
175 PTAS
77
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VII-Fascículo 77

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa. Santa Perpètua de Mogoda (Barcelona) 038507
Impreso en España-Printed in Spain-Julio 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.





Necesidades vitales

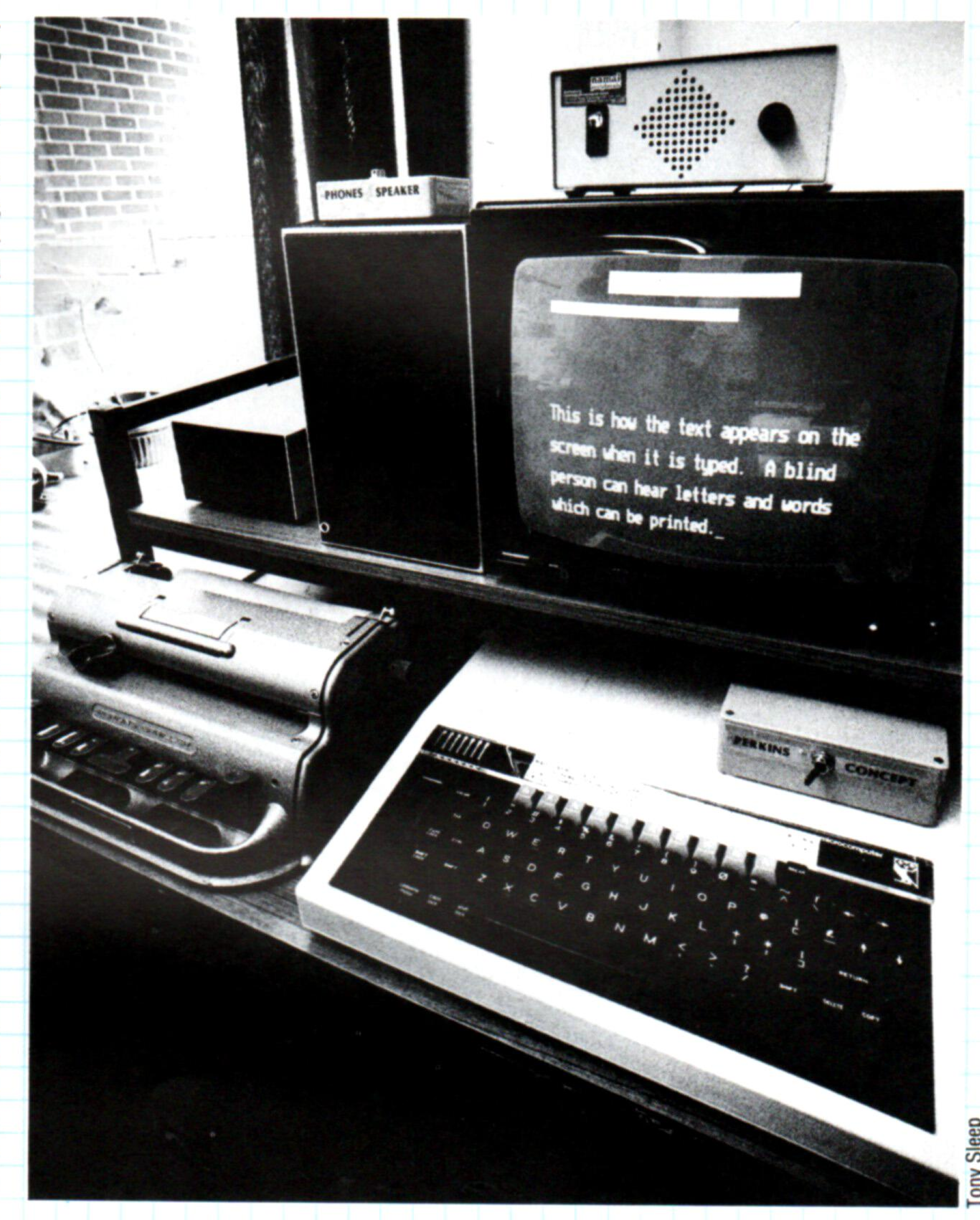
Para los minusválidos el microordenador constituye un medio vital para superar las barreras del mundo de la normalidad

Muchos niños y adultos tienen necesidades especiales desde el punto de vista de la educación. Ello puede deberse a una minusvalía física como la sordera o la ceguera, o a una deficiencia mental. Si bien hay algunos requerimientos que son obvios, hay otros que son difíciles de diagnosticar y comprender y cada incapacidad presenta sus propios problemas, a menudo originados por la insensibilidad de la sociedad más que por la incapacidad en sí.

Lamentablemente, los recortes presupuestarios practicados por algunos gobiernos inciden tan negativamente en las pocas escuelas especiales que existen como en las de educación general. Por consiguiente, éste no es un mercado lucrativo para quienes desarrollan hardware y software. No obstante, la energía y la dedicación de quienes trabajan en este campo pueden servir de ejemplo para el resto del sistema educativo. Un grupo de voluntarios de South Yorkshire, en Gran Bretaña, por ejemplo, reunió 10 000 libras esterlinas en su tiempo libre para desarrollar un teléfono especial de bajo costo para los sordos. Gran parte de este tipo de trabajo lo realizan a nivel de "comunidad" personas individuales y grupos de voluntarios, y a lo largo y ancho de las islas hay maestros de escuela, programadores e ingenieros que están trabajando por iniciativa propia, desarrollando hardware y software para educación especial.

En Gran Bretaña se han abierto SEMERC (*Special Education Microelectronics Resources Centres*: centros de recursos microelectrónicos para la educación especial) en Manchester, Newcastle y Redbridge, para satisfacer las necesidades de la educación especial. Una de sus principales funciones es educar a los maestros de escuelas especiales e informarles de los desarrollos pertinentes. Los recursos de los cuatro SEMERC son pequeños y su labor es muy amplia. El centro de Redbridge atiende a 700 escuelas y debe dividir su tiempo entre educar al personal, dar consejos e información, y evaluar y desarrollar hardware y software.

La situación se complica aún más por los muchos problemas diferentes de la educación especial. En muchas incapacidades, por ejemplo, es la comunicación lo que ofrece los problemas más graves. Una persona inteligente y vivaz puede estar atrapada en un cuerpo incapacitado, imposibilitada de comunicarse con el mundo exterior. La microtecnología ha abierto nuevos canales de comunicación para muchas personas. El Photonic Wand, por ejemplo, es un sensor óptico montado en un casco plástico y controlado por los movimientos de la cabeza, y permite que las personas que no pueden hablar y no tienen control sobre sus extremidades puedan operar un BBC Micro. El aparato se conecta al ordenador a través de la puerta para entrada analógica. El sensor óptico es similar a un lápiz óptico y mueve un cursor a través de la pantalla en respuesta a los movimientos de la cabeza.



Tony Sleep

Lugar de trabajo
El puesto de trabajo Vincent combina numerosas modificaciones que lo hacen muy útil para personas con diversas incapacidades. El Perkins Brailler permite que los invidentes entren información en el ordenador, y el sintetizador de voz proporciona la realimentación audible. La visualización de textos en la VDU se ha agrandado para beneficiar a los parcialmente invidentes, y el puesto de trabajo también está conectado a un teclado conceptual. Si bien el puesto de trabajo reflejado aquí tiene acoplada una impresora normal, también existe una impresora Braille especial, pero su precio la hace poco asequible

Con este aparato se puede usar un programa para tratamiento de textos denominado *Write*. Visualiza el alfabeto en la pantalla y pueden seleccionarse letras individuales, en mayúscula o minúscula, señalándolas con el cursor. Dispone de una función de edición y de listas de palabras. El texto se puede imprimir, salvar o suprimir. Un programa llamado *Paint* permite dibujar con seis colores, y otro, llamado *Music*, visualiza un teclado. Las notas a tocar se seleccionan con el cursor.

Este dispositivo lo pueden utilizar personas con movimiento inestable de la mano y niños a partir de ocho años. En el SEMERC de Manchester se han llevado a cabo trabajos para conectar la Photonic Wand a un sintetizador de voz. Se seleccionan de la pantalla respuestas diseñadas para conversaciones

telefónicas, tales como "¿puedes repetir esa frase?", que son pronunciadas por un sintetizador de voz de D.E. Systems. Se espera ampliar el vocabulario y desarrollar una forma sencilla de conectar el sintetizador al teléfono.

Los ordenadores se diseñan para las personas que pueden ver. Las pantallas, las impresoras y los plotters (trazadores de gráficos) son todas salidas visuales y la totalidad de los programas se valen de las visualizaciones. El proyecto Computing and the Blind (La informática y los ciegos) de la Open University ha estado investigando formas de adaptar el hardware de bajo costo existente para que lo utilicen los invidentes. El proyecto se ha puesto en marcha en colaboración con los ciegos en la escuela, en el trabajo y en casa. En las escuelas y centros laborales se han instalado puestos de trabajo que constan de un microordenador BBC, unidad de disco, pantalla, impresora, sintetizador de voz, teclado conceptual y un Perkins Brailler modificado.

El Perkins Brailler se inventó hace 40 años para mecanografiar en braille, sistema de escritura que consiste en un código de puntos repujados sobre papel que permite la lectura al tacto. Se ha conectado en interface con el ordenador para que un invidente pueda leer la salida impresa. Se dispone de software para que el ordenador transcriba el braille a texto normal y pueda almacenarlo, editarlo e imprimirlo.

Se han desarrollado varios procesadores de textos parlantes para ser usados con el teclado estándar. Los caracteres digitados y las teclas especiales se verifican mediante la salida hablada. La tecla Delete también pronuncia el nombre del carácter borrado. El texto se edita utilizando un cursor de voz. A medida que éste se va desplazando a través del texto, se van pronunciando los caracteres, palabras o frases. El cursor se puede detener para añadir o suprimir texto. Se definen márgenes, cabeceras y espacios para producir una copia final pulcra, imposible de diferenciar de una producida por una persona vidente.



Un toque mágico
La banda fotónica permite que incluso personas gravemente incapacitadas, imposibilitadas para hablar y mover sus extremidades, puedan controlar un BBC Micro. El movimiento de la banda se detecta mediante un sensor óptico y se traduce a una señal analógica

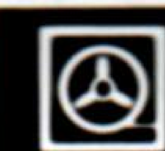
Chris Barker, cortesía de Educational Computing Magazine

Algunos niños incapacitados no poseen la suficiente coordinación física como para utilizar un teclado estándar. Pensando en estas personas se ha desarrollado un teclado plano sensible al tacto, denominado *teclado conceptual*, que permite colocar encima cubiertas intercambiables diseñadas por el maestro. El teclado, por ejemplo, se puede dividir en cuatro secciones y dibujar una imagen en cada cuarto, posibilitando el control de una tortuga para el suelo. Pulsando diferentes secciones se puede dirigir la tortuga hacia adelante, atrás, izquierda y derecha. Lo cual es más sencillo que tener que digitar FD 1 RETURN. Para el teclado conceptual existe algo de software y un lenguaje denominado Starset.

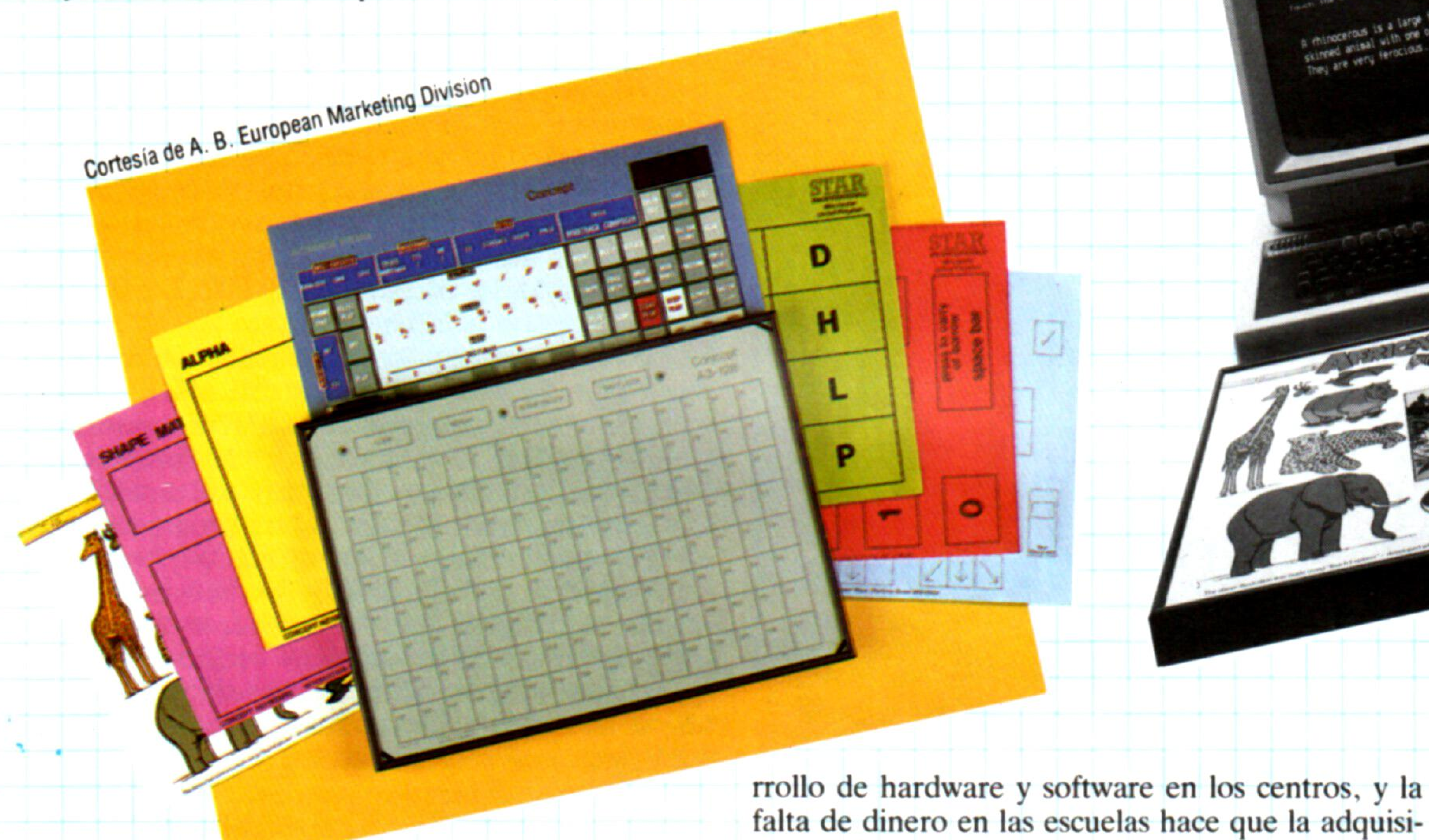
El "ratón" que se utiliza con el Apple Macintosh también evita la "barrera del teclado" y posee un enorme potencial para la educación especial.

Se han desarrollado interruptores especiales, operados mediante diversas partes del cuerpo. Por ejemplo, se ha diseñado un interruptor para personas que carecen de movimiento voluntario. Dos pequeños discos metálicos colocados sobre la piel junto a los ojos detectan el movimiento horizontal del globo ocular y las señales eléctricas se amplifican sobre unos interruptores de control.

En el Departamento de Ciencia y Sociedad de la Universidad de Bradford se le instaló a una tortuga una luz naranja intermitente, que podía ser vista por un incapacitado físico de 18 años de edad. Utilizando un teclado conceptual, el adolescente fue capaz de desplazar la tortuga por el suelo y ver los resultados de sus acciones. Para este adolescente, así como para muchos niños incapacitados, experiencias como ésta suelen representar el primer paso para dejar de ser un observador pasivo de su entorno y comenzar a influir conscientemente en el mismo.



En el congreso del British LOGO User's Group celebrado en 1984, la doctora Sylvia Weir ofreció notables ejemplos del progreso que experimentaron niños incapacitados mental y físicamente gracias al empleo del LOGO y las tortugas. Michael, de 17 años y alumno de una escuela especial, jamás había escrito ni una palabra. Sus maestros intuían que era inteligente y que nunca había tenido ocasión de demostrarlo. Tras introducirlo en la informática, al poco tiempo se pasaba diez horas diarias programando. Dos años después estaba en la universidad escribiendo un trabajo titulado *Inteligencia prisionera*. A un niño autista de siete años de edad se le permitió controlar una tortuga desde una caja de botones, un dispositivo similar al teclado conceptual. Se mostró tan entusiasmado con la experiencia, que por primera vez comenzó a hablar.



Idea conceptual
El teclado conceptual responde al principio de la tablilla de gráficos. El bajar la resolución del tablero y añadir una cubierta permite que incluso los usuarios con falta de coordinación física puedan comunicarse eficazmente con el ordenador sin tener que dominar un teclado QWERTY. El diseño de las cubiertas puede ayudar a reducir la renuencia del nuevo usuario a acercarse al ordenador

La doctora Weir también ha empleado el LOGO en una escuela especial con un grupo de adolescentes afectados de parálisis cerebral. Ha escrito: "La notable mejora resultante, en términos del sentido de la valía personal y las consecuencias de la actividad intelectual, han llevado a la escuela a implementar su propio centro de informática, utilizando un sistema de ordenadores recientemente implementado."

Un esfuerzo conjunto del SEMERC de Manchester y un club de informática local produjo el Micromike, un dispositivo para ayudar a los niños con defectos de habla. Un micrófono de radio CB modificado que se conecta a un microordenador BBC, permite al niño controlar diversas actividades en la pantalla utilizando su voz. El software desarrollado para el mismo incluye *City*, programa que permite dibujar la línea del horizonte de una ciudad. La altura y la anchura de los edificios se determinan mediante el volumen y la duración de la voz. Otros programas permiten representar estrellas en el cielo, conducir un barco a través de los rápidos y conseguir que un helicóptero rescate a un piragüista. Todos ellos le ofrecen al niño la oportunidad de controlar su voz de una forma única.

Al igual que la educación normal, la educación especial está a la espera del desarrollo tecnológico. Carece de fondos como para comprar y en el mundo comercial es muy poca la investigación y el desarrollo que se realizan para ella, porque las casas de software y hardware ganan dinero con las empresas y las industrias y no con la educación, y el mercado de la educación especial es aun más reducido. Los centros SEMERC no pueden hacer frente a las demandas actuales de ayuda y la cantidad de escuelas especiales por cada área es inmensa. La escasez de fondos y de experiencia retrasa el desarrollo de hardware y software en los centros, y la falta de dinero en las escuelas hace que la adquisición de la tecnología de ordenadores sea lenta. No obstante, a pesar de esta carencia de recursos, se está llevando a cabo un trabajo apasionante.

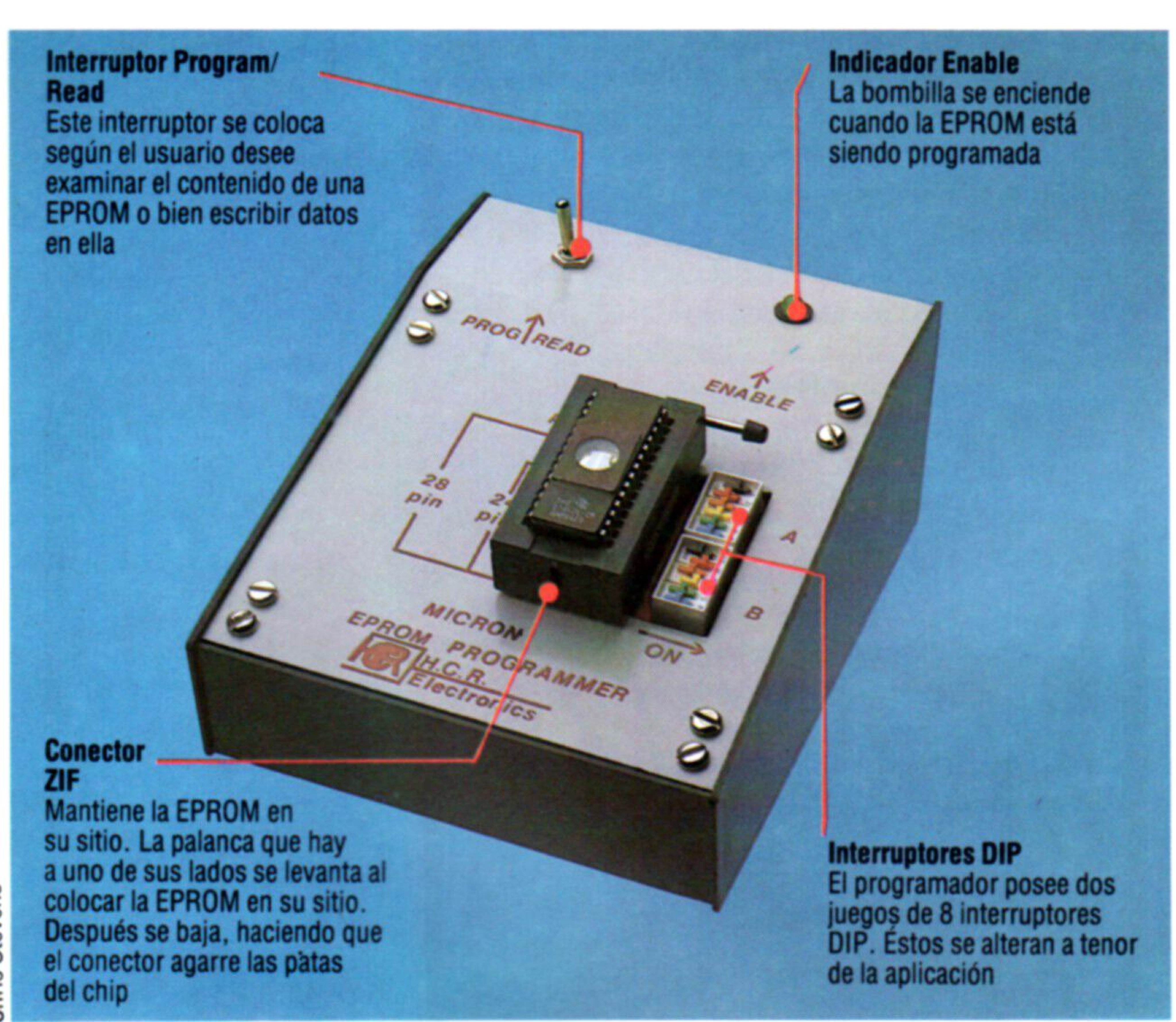
Informática y genio

Los niños dotados y de talento también tienen necesidades educativas especiales y el empleo de micros en la docencia les proporciona un medio de expresión tan interesante como lo es para los incapacitados. De hecho, muchos niños con incapacidades físicas o de percepción están muy dotados intelectualmente.

Tal como sucede con los incapacitados, el micro les proporciona a los superdotados un medio de ganar control sobre el entorno y superar las barreras naturales o impuestas por el hombre hacia el crecimiento individual y la autorrealización. Los ordenadores le ofrecen al niño superdotado:

- La oportunidad de desarrollar aptitudes e intereses a su propio ritmo;
- Formas nuevas y creativas de enfocar la resolución de problemas;
- Un medio para comunicarse y cooperar con otros estudiantes dotados;
- La oportunidad de practicar ejercicios reiterativos y agotadores de una forma que impide el aburrimiento y mantiene su interés;
- Un nuevo método (y campo) de descubrimiento.

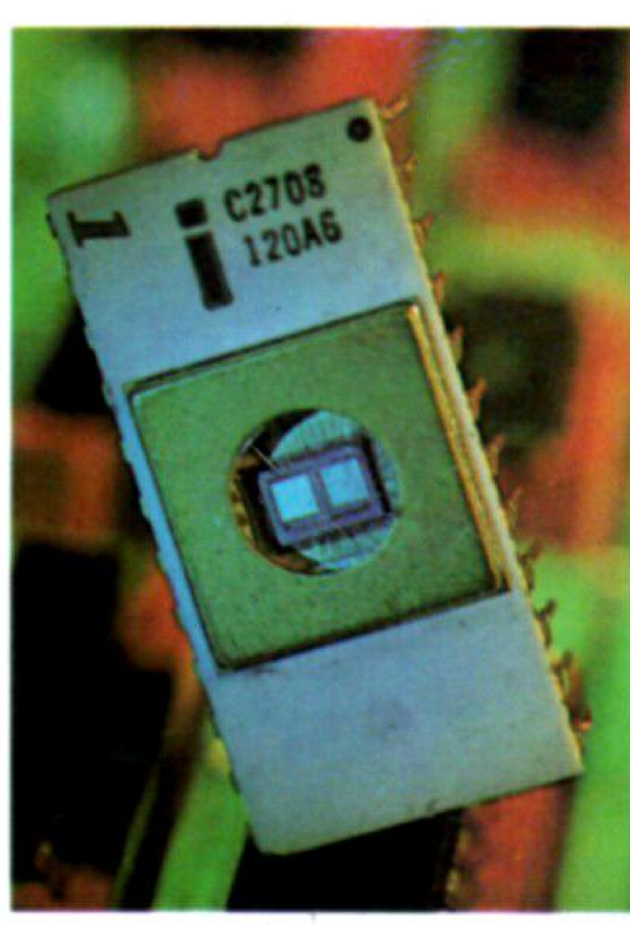
Veloz y eficiente



Chris Stevens

Componentes del grabador
Un programador de EPROM permite que los usuarios tengan sus programas permanentemente en la memoria. Los programas se cargan en el ordenador y después se transfieren al programador a través de la puerta para el usuario, donde se "soplan" en el chip

En esta ocasión nos referiremos al Micron, dispositivo económico y eficaz para programar memorias EPROM



Mirándolo de cerca
El chip que hay dentro de la EPROM se puede observar fácilmente a través de la ventana de cuarzo del centro. Al examinarlo con detenimiento se podrán apreciar los delgados trozos de cable que unen las conexiones de E/S con las patillas y las matrices de que se compone la EPROM

Uno de los mayores méritos del BBC Micro es su flexibilidad. Al diseñar la máquina, Acorn optó por incluir cuatro ranuras para ROM adicionales, permitiendo que el usuario colocara unidades ROM extras para aplicaciones específicas en la máquina, y éstas se pueden "paginar" mediante la instrucción*. La inclusión de estas ranuras ha sido uno de los mayores éxitos de la máquina, dado que permite disponer de programas instalados en la placa a los que se puede acceder con mucha más rapidez que si se hubiera de cargar de cinta o disco.

En la actualidad existe a la venta una enorme selección de estas ROM de aplicaciones, abarcando una amplia gama de programas, incluyendo tratamiento de textos, bases de datos y lenguajes como el LOGO. Sin embargo, los usuarios también pueden escribir sus propios programas y transferirlos luego a una EPROM, que permite almacenarlos con carácter permanente en la máquina. Esto se consigue programando la EPROM con un dispositivo al que, como es natural, se denomina *programador de EPROM* y que popularmente se conoce como *soplador de EPROM*.

Al igual que todas las ROM, una EPROM se compone de una matriz de líneas eléctricas en columnas y filas, cada intersección de las cuales se denomina *celda*. De haber una conexión entre una línea de una columna y una de una fila, el estado lógico de ese punto será uno; de lo contrario, será cero. Cuando llega al bus de direcciones una serie de impulsos eléctricos correspondientes a una dirección, la electricidad pasará por ocho de estas líneas (estos ocho impulsos representan un byte). Según haya o no una conexión en las líneas de cruce apropiadas, la carga eléctrica pasará hasta las líneas de cruce y determinará el envío de una carga, a través del bus de datos, de vuelta al procesador. De este modo, los datos retenidos en una dirección se transfieren a la CPU.

Los estados lógicos en la matriz de memoria de una EPROM suelen estar establecidos en uno: cada dirección contiene el valor &FF. Si a través de una celda pasa un gran voltaje, la conexión se rompe produciendo por consiguiente el estado lógico complementario de cero. Para programar una dirección determinada de una EPROM, hemos de establecer primero el patrón de bits necesario en las patillas de dirección, para especificar la dirección a programar. Luego enviamos el código requerido por el bus de datos. A las patillas de datos apropiadas se les aplica un impulso de 50 microsegundos a 12 V, y se programa la dirección. Incrementando la dirección y enviando otro patrón de bits a las patillas de datos, se puede programar toda la EPROM. No obstante, a diferencia de otros tipos de ROM programables, en las que este proceso es irreversible, una EPROM, tal como indica su nombre, se puede borrar y programar de nuevo varias veces.

Una EPROM se caracteriza por una pequeña ventana de cuarzo que hay en la parte superior del chip. Cuando a través de esta ventana pasa una intensa luz ultravioleta, los átomos del chip se ionizan en un nivel eléctrico superior y las uniones entre los átomos se vuelven a establecer. Ello permite que la corriente pase una vez más entre las líneas de cruce de la matriz, y el estado lógico vuelve a ser uno. La luz ultravioleta, sin embargo, no es estrictamente necesaria, puesto que el programador de EPROM puede escribir una rutina de software para restaurar todas las posiciones de memoria a &FF.

El programador Micron EPROM, de HCR Electronics, le permite al usuario programar hasta 16 K de código máquina en una EPROM. El programador es una caja pequeña, cuya característica más destacada es un conector ZIF (*Zero Insertion Force*: fuerza de inserción cero). Éste permite insertar las EPROM sin dañar a las patillas que, una vez rotas, dejan al chip inutilizado. El conector ZIF está diseñado para alojar chips de 24 o 28 patillas. Junto al conector ZIF hay dos juegos de interruptores DIP, que se manipulan según cuál de las diversas opciones se utilice. En la parte superior izquierda del programador hay un interruptor que es nece-

sario posicionar en función de si se desea programar la EPROM o leerla. Obviamente, se ha de tener cuidado cuando se utiliza el programador, asegurándose de que el interruptor esté en la posición correcta, de lo contrario la EPROM se podría volver a "soplar", perdiéndose en consecuencia el programa almacenado en el chip.

El dispositivo posee un cable plano que se conecta a la puerta para el usuario del BBC desde la fuente de alimentación eléctrica. Uno puede cargar y salvar programas en un área de buffer y luego transferirlos a la EPROM con el software activado por menú que se suministra con la máquina. El software está almacenado en cassette y, como es natural, es aconsejable que lo primero que haga el usuario sea volcarlo en una EPROM.

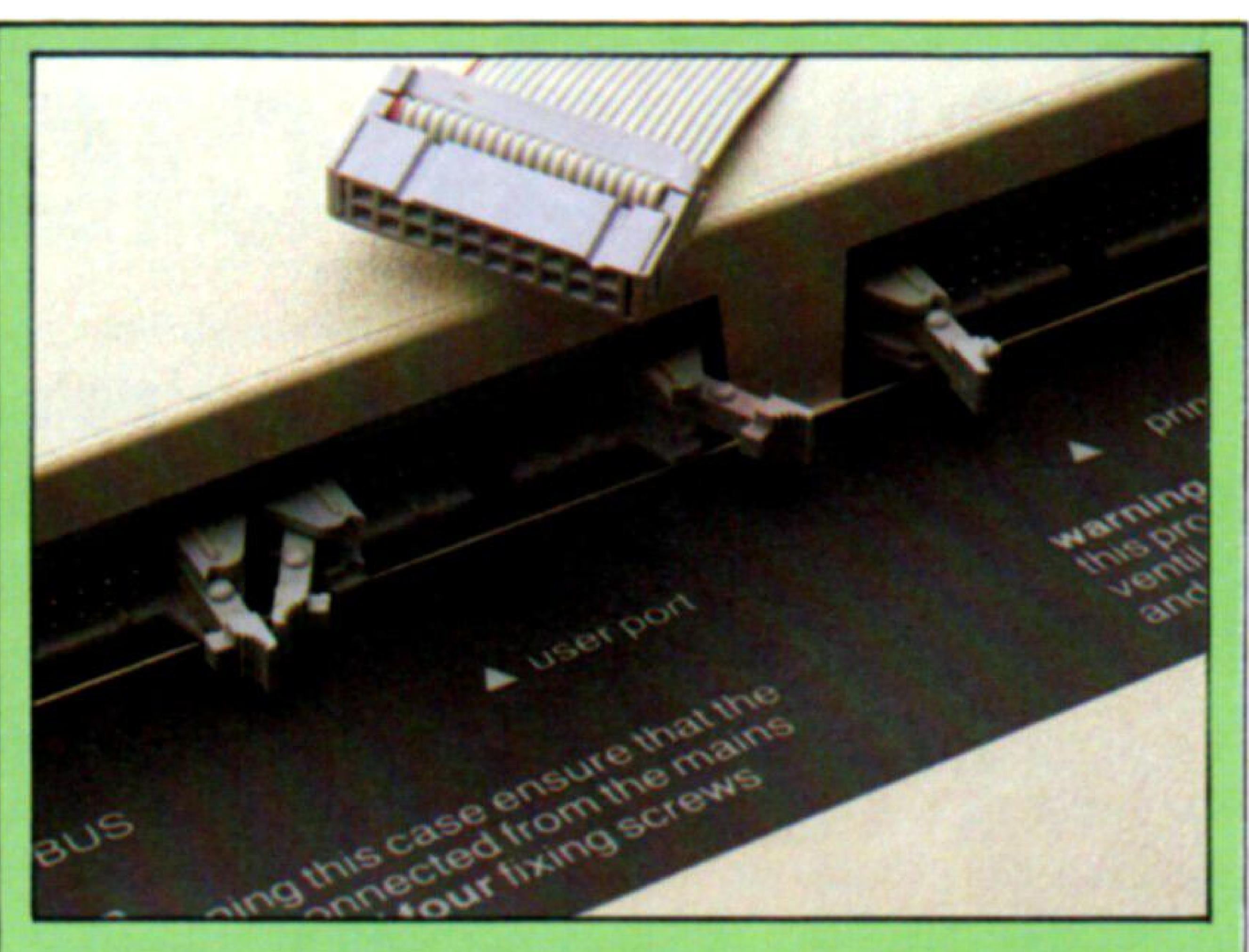
Una vez cargado, el software presenta un menú en la pantalla ofreciendo las opciones disponibles. Para copiar un programa en una EPROM, primero se debe cargar éste en la memoria utilizando la opción (L)oad (cargar). Esta instrucción tiene el efecto de colocar el programa en un área de buffer, reservada por el software EPROM, que comienza en la posición &2000. Tras haber hecho esto, el usuario selecciona la opción (P)rograma del menú principal. La pantalla visualiza entonces la lista de tipos de EPROM que se pueden programar mediante este dispositivo. Después de escoger una EPROM, la pantalla visualizará los 16 interruptores DIP y sus posiciones; los interruptores que es necesario modificar tras el programa anterior destellarán de forma intermitente. El programa solicita al usuario que mueva el interruptor de lectura/escritura a PROG. Pulsando cualquier tecla se inicializará la programación de la EPROM, visualizándose cada dirección a medida que va siendo programada.

El período de tiempo que lleva el proceso de "soplado" depende del número de direcciones de la EPROM que no vayan a ser modificadas. Ello puede consumir entre 74 segundos para un programa corto y un máximo de 14 minutos para una EPROM de 16 K. Mediante el empleo de la instrucción (V)erificar, uno puede comprobar si la EPROM se ha programado correctamente. El software calcula una suma de control, y las posibles direcciones defectuosas se listan en la pantalla. De no haber ningún error, la EPROM se puede instalar con toda seguridad en uno de los conectores para ROM del BBC Micro.

A los programas se les puede dar nombres normales. Utilizando la instrucción G, es posible agregar al comienzo del programa una rutina, asociando un nombre, al que luego se podrá llamar con el prefijo * y una instrucción CALL. Cuando se cargan en la EPROM programas en BASIC, esta instrucción se emplea junto con la opción (F)*ill* (llenar). Esta instrucción asegura que las direcciones no utilizadas por el programa en BASIC permanezcan establecidas en &FF y continúen, por tanto, siendo programables. A los programas en código máquina también se les puede dar nombres, pero éstos sólo se pueden cargar desde la dirección &2000.

Considerando lo útiles que son estos dispositivos, quizá sea sorprendente que los programadores de EPROM no hayan adquirido una mayor popularidad entre los usuarios del BBC Micro. Es probable que muchos de ellos crean que estos programadores están reservados para los amantes más "serios" de la electrónica y, por tanto, fuera del alcance de la mayoría de los usuarios de ordenadores personales. Esto no es así. Los programadores de EPROM son fáciles de usar, no requieren ningún conocimiento de electrónica, y las máquinas ofrecen numerosas ventajas. Entre éstas, una de las principales es la velocidad de acceso: se puede llamar instantáneamente a un programa a través del sistema operativo sin tener que cargarlo de disco o cinta. Además, una vez que los programas se incorporan al ordenador, las aplicaciones retenidas en la EPROM se pueden llamar desde programas en BASIC.

Por último, los usuarios disfrutan de la inmensa satisfacción de ver que los programas que ellos han escrito ocupan un lugar entre los programas incorporados del BBC Micro.

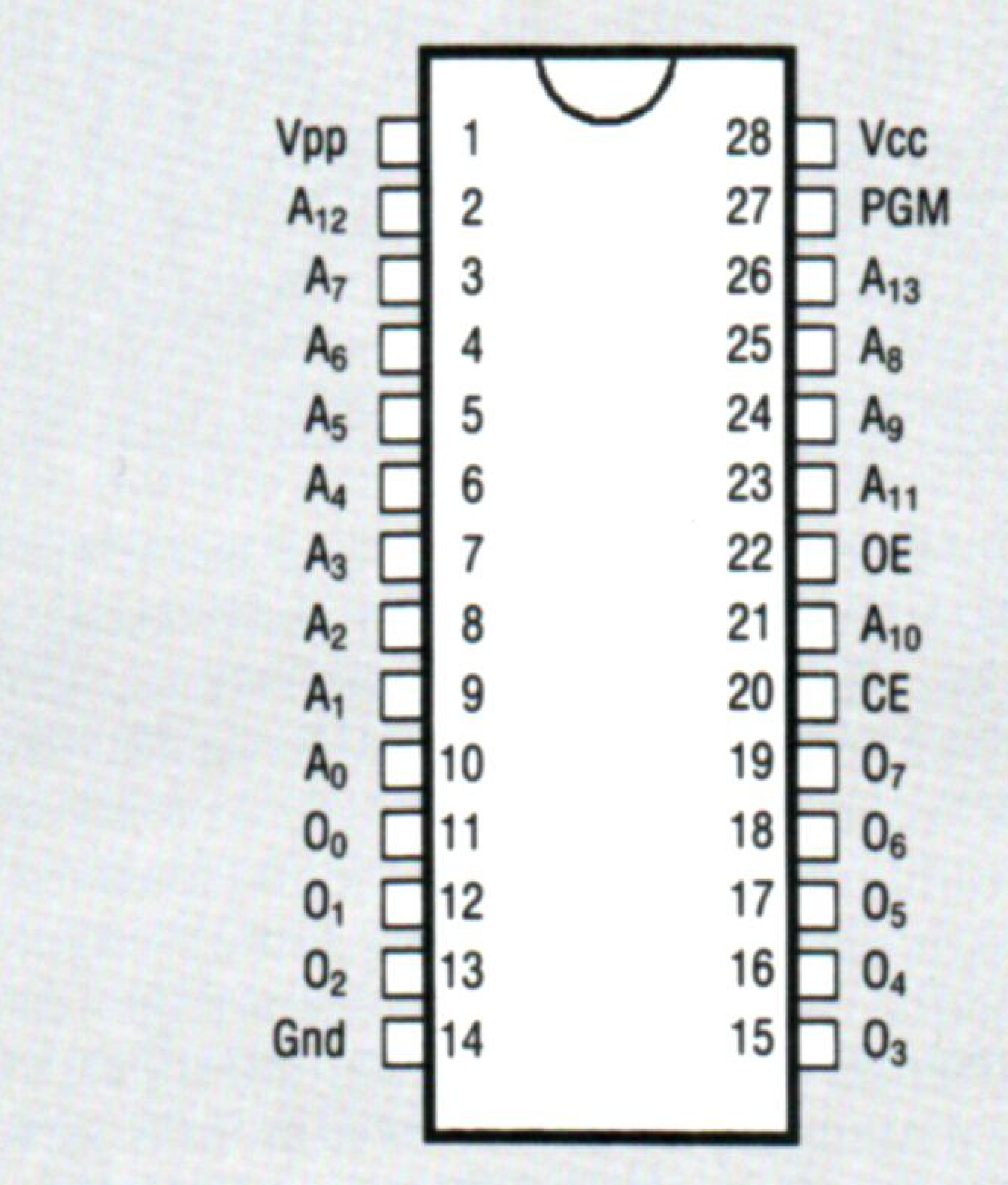


Conexiones de interface
El programador de EPROM se conecta, a través de un cable plano de 12 vías, a la puerta para el usuario del BBC Micro

Precisiones

Nombres de las patillas	
A_0-A_{13}	Direcciones
CE	Chip Enable (habilitación chip)
OE	Output Enable (habilitación salida)
O_0-O_7	Salidas
PGM	Programa
Vpp Vcc	Voltajes de alimentación
Gnd	Tierra

He aquí una representación de la EPROM de 16 K TMS27128, uno de los chips que se pueden "soplar" utilizando el programador HCR. Se trata de un dispositivo de 28 patillas (si bien también acepta chips de 24). El chip posee 14 patillas de dirección que le permiten direccionar los 16 K de memoria. El TMS 27128 posee, asimismo, 8 patillas para proporcionar una salida de 8 bits



Liz Dixon

EPROM BLOWER

DIMENSIONES
140×105×65 mm

INTERFACES
Conector de cable plano que se enchufa en la puerta para el usuario del BBC Micro

DOCUMENTACION
Tres hojas fotocopiadas que explican cada una de las nueve instrucciones de que dispone la máquina. Aunque la máquina es fácil de utilizar, el tono general de la documentación es un tanto elevado para el principiante

VENTAJAS
La máquina es fácil de usar. El software activado por menú no requiere por parte del usuario ningún conocimiento de la mecánica de la máquina para poder operarla

DESVENTAJAS
Poca explicación acerca de cómo podría desarrollar el usuario un uso de la máquina más allá de seguir las instrucciones del software. No es fácil transferir a EPROM los programas en código máquina que no se puedan volver a compilar para hacer que empiecen en &2000

Para decidir

Ahora nos ocuparemos de las construcciones para toma de decisiones: las sentencias IF y CASE

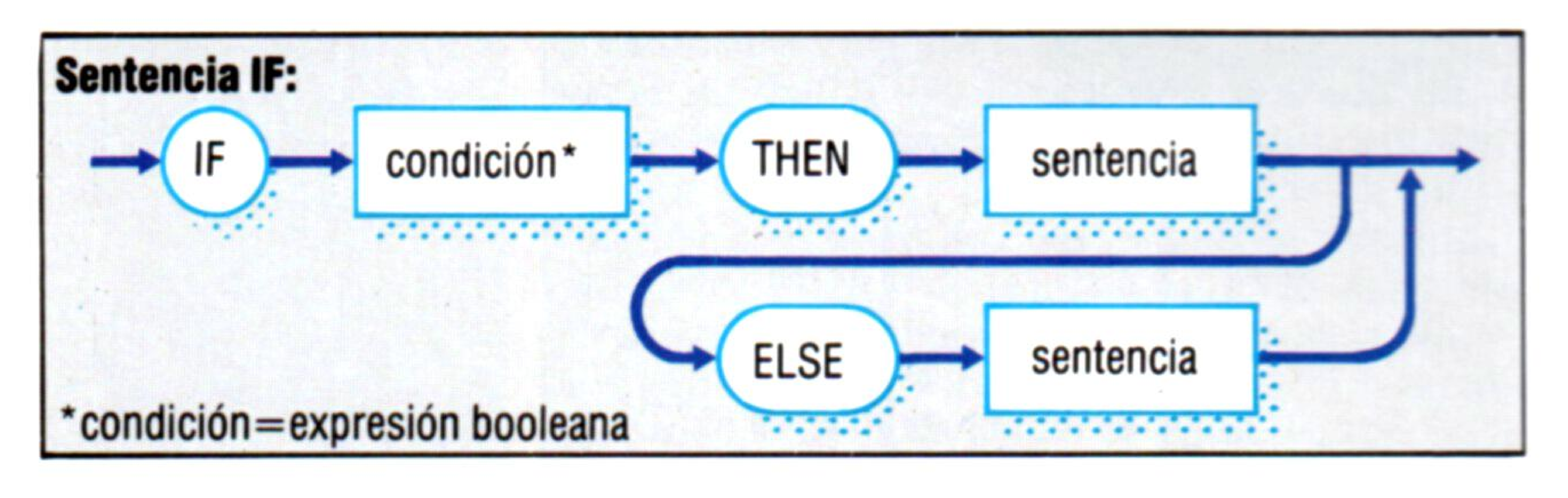
La sentencia IF del PASCAL es similar a la de la mayoría de los lenguajes. Dado que con el final de la línea no concluye una sentencia, podemos utilizar el formato libre del PASCAL para mostrar los posibles caminos a través de la estructura de la sentencia IF con indentación lógica. Éstas son dos sentencias IF:

```
IF contador=límite
  THEN
    WriteLn('No hay lugar')
  ELSE
    write('Siguiente?');
IF número>máximo THEN
  máximo:=número
```

En el segundo ejemplo, la ausencia de una cláusula ELSE implica:

```
  ELSE
    {no hacer nada}
```

La alineación de las palabras reservadas THEN y ELSE (cuando está presente) ayuda a que el ojo pueda seguir el flujo del control a través de la "construcción". Se utiliza un punto y coma final para separar sentencias, tal como vemos en el ejemplo, y por lo tanto *jamás* debe aparecer antes de un ELSE. Recuerde que en PASCAL la sentencia ELSE no existe, sólo existe la sentencia IF; THEN y ELSE se emplean para delimitar la condición booleana y las sentencias de las dos cláusulas. En realidad, cada vez que utilizamos una sentencia IF estamos comprobando un valor booleano. La cláusula THEN sólo se ejecuta si la expresión evaluada es verdadera y, de seguir una cláusula ELSE, la(s) sentencia(s) en ella especificadas se llevarán a cabo si el valor booleano es falso.



El PASCAL fue el primer lenguaje que introdujo el tipo de datos conceptual conocido como escalares enumerados. Éstos son sumamente útiles, dado que nos permiten mantener una visión de alto nivel respecto a la clasificación de los datos, sin tener que traducir continuamente de forma mental los datos a códigos numéricos. (Los números son el lenguaje del ordenador, no el nuestro... a menos que estemos resolviendo un problema de naturaleza específicamente matemática.) Ya vimos cómo nos puede ser de ayuda una simple definición de constante:

```
CONST
  anchura=80;
```

El identificador de constante anchura se puede entonces utilizar a lo largo de todo el programa en referencia al número de columnas de la pantalla o la impresora. Reescribir el programa para una VDU de 40 columnas es entonces una cuestión tan sencilla como cambiar una línea de la definición del principio del programa; todos los cálculos para el formateo de la impresión, etc., se alterarán, en consecuencia, automáticamente. De este modo, se podría entonces definir una escala completa de constantes.

Si estuviéramos empleando gráficos en color, los colores disponibles se asociarían a una escala adecuada de números (rojo=1, verde=2, p. ej.), ¡pero esto teóricamente se podría utilizar entonces calculando la raíz cuadrada de verde, o multiplicando azul y amarillo! Esto no sólo es ilógico, sino que también constituye una potencial fuente de error cuando estamos razonando problemas que en sí mismos no implican datos numéricos. El elemento de definición TYPE de un programa en PASCAL se puede utilizar para definir un tipo escalar conceptual totalmente nuevo, simplemente enumerando una lista de identificadores que representen a todos los valores de constantes de la escala. Por ejemplo:

```
TYPE
  tinte=(rojo,verde,amarillo,azul,magenta,cyan);
```

Puesto que se trata de una definición (del nuevo tipo) y no de una declaración, la sintaxis emplea el signo de igualdad para definir el identificador del tipo (tinte) para referirse a los valores ordenados del tipo encerrado entre paréntesis. De forma análoga, el tipo predefinido "entero" alude a todos los números enteros disponibles en la implementación del PASCAL.

Como sucede siempre en PASCAL, los sucesivos identificadores de una lista (en este caso, valores de color) se separan entre sí mediante comas. Estos valores, por supuesto, están asociados internamente a valores enteros, con la numeración ordinal comenzando por cero. Esta representación numérica la organiza el compilador de forma automática, y es muy similar a los códigos implícitos para el juego de caracteres del ordenador. Así como el carácter A posee un código ASCII de 65, cada valor de tinte tendrá un número ordinal que podemos obtener mediante la función escalar ord. Por lo tanto, en este ejemplo, ord (rojo) es 0 y ord (cyan) sería 5. Ahora, habiendo definido este nuevo tipo escalar, podríamos declarar una variable en la forma habitual:

```
VAR
  color : tinte;
```



Esto declara un identificador (color) como un elemento de datos del tipo tinte, tal como la declaración:

```
VAR
  letra : char;
```

indica la naturaleza de carácter del objeto de datos denominado letra. Las únicas operaciones definidas sobre tipos de enumeración son las comparaciones de relación y el empleo de funciones escalares. Por ejemplo, podríamos escribir:

```
if color<cyan then
  color:=succ(color)
```

¡Recuerde que pred(rojo) y succ(cyan) no existen! La variable color es incompatible con variables de cualquier otro tipo, escalares u otras. Ello significa que ya no podemos efectuar funciones ilegales, como extraer raíces cuadradas o decir:

```
color:=color+1
```

Esto conduce a una restricción evidente. Los caracteres y los números se pueden utilizar como parámetros para sentencias write y WriteLn, pero

```
WriteLn(color)
```

sería ilegal. Ello se debe a que los valores del tipo son puramente conceptuales y si deseamos imprimir sus nombres, debemos asociar los valores de color a series de caracteres. Ésta es una aplicación ideal para la otra construcción de opciones del PASCAL, la sentencia CASE.

Hemos visto lo familiar que nos resulta la sentencia IF del PASCAL y los beneficios que suponen las convenciones de libre formato del PASCAL desde el punto de vista de la legibilidad. Hay ocasiones, sin embargo, en las que se han de tomar decisiones de múltiples opciones que requerirían algo como:

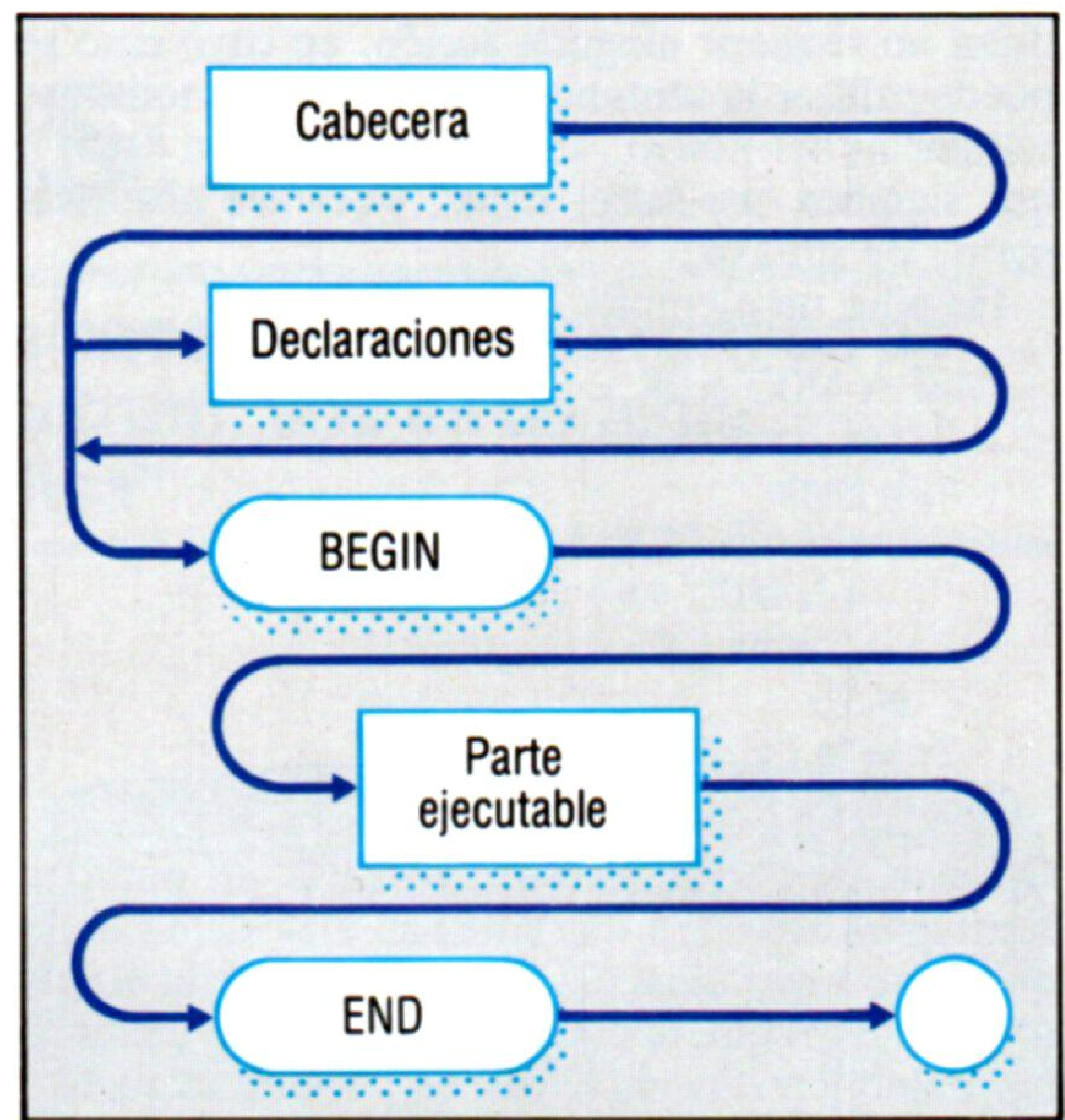
```
IF N=1
  THEN
    write ('primero')
  ELSE
    IF N=2
      THEN
        write ('segundo')
      ELSE
        IF N=3
          THEN
            write ('tercero')
          ELSE
            write ('mayor de tres')
```

Siempre que la sentencia a ejecutar dependa de que el valor de una variable escalar simple esté comprendido en una gama limitada de valores podemos utilizar la sentencia CASE por razones de comodidad. En este caso:

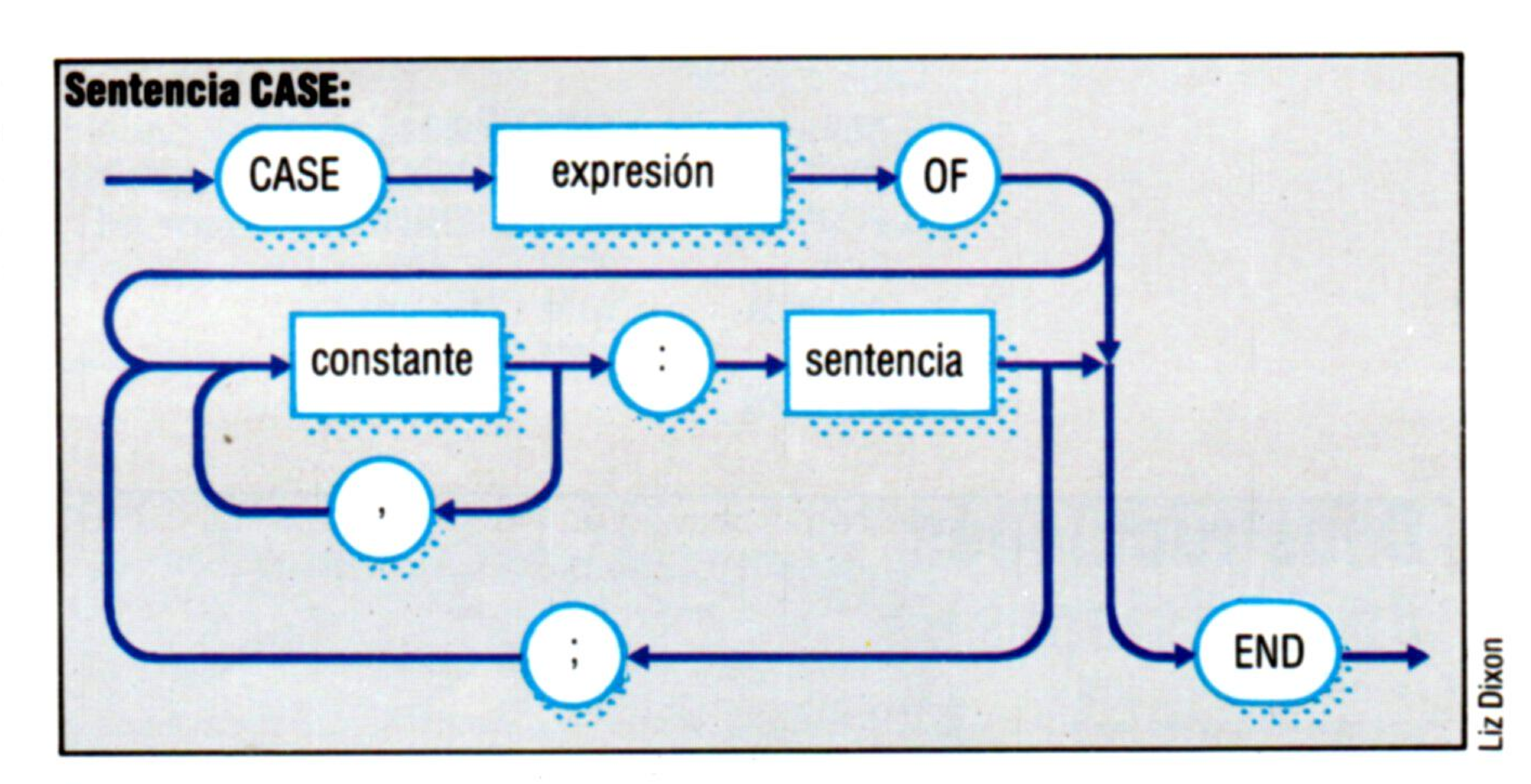
```
CASE N OF
  1          :write ('primero');
  2          :write ('segundo');
  3          :write ('tercero');
  4,5,6,
  7,8,9      :write ('mayor de tres')
END  {CASE}
```

Observe que esto sólo es satisfactorio para valores de N comprendidos en la escala de "etiquetas de CASE" especificada en el "cuerpo" de la sentencia CASE: entre 1 y 9, en este ejemplo. Todos los valores que pudiera tener N en tiempo de ejecución se *deben* especificar individualmente y sería ilegal, por ejemplo, que se entrara N=0.

Muchos compiladores de PASCAL poseen la palabra reservada adicional OTHERWISE u OTHERS, que se puede emplear para especificar una acción por defecto. Remítase a su manual para la sintaxis necesaria en este caso.



Esqueleto de un programa
Éste es el diagrama sintáctico simplificado de un programa en PASCAL, que muestra los principales elementos. Observe que las palabras reservadas del PASCAL aparecen en mayúsculas. Ésta es una convención que se ha desarrollado para diferenciarlas de las palabras no reservadas. La mayoría de las versiones de PASCAL, sin embargo, aceptarán palabras reservadas en minúsculas



Ésta es la única sentencia del PASCAL que utiliza END para delimitar una estructura que no se haya entrado mediante un BEGIN, de modo que es una práctica común, y de buen estilo, calificar cada END de una sentencia CASE tal como se indica.

La construcción trabaja evaluando la expresión escalar delimitada por las palabras reservadas CASE y OF. Un nombre de variable simple es una expresión mínima que no requiere cálculo. El valor obtenido se compara entonces con cada una de las constantes especificadas en las listas de etiquetas del CASE, y cuando se halla una pareja se ejecuta la sentencia que sigue a los dos puntos, y *sólo* esa sentencia. El flujo de control, por consiguiente, no se desvía, de modo que se mantiene la integridad estructural de la construcción. Si fuera necesario llevar a cabo varias operaciones, se podría utilizar, por supuesto, una sentencia compuesta encerrada entre un par BEGIN/END.

En ciertas circunstancias, algunos valores po-

drían no requerir ninguna acción, en cuyo caso se puede utilizar la sentencia más simple de todas las sentencias del PASCAL. Ésta es la sentencia "nula", que significa "no hacer nada" y ¡carece absolutamente de sintaxis!

He aquí un ejemplo:

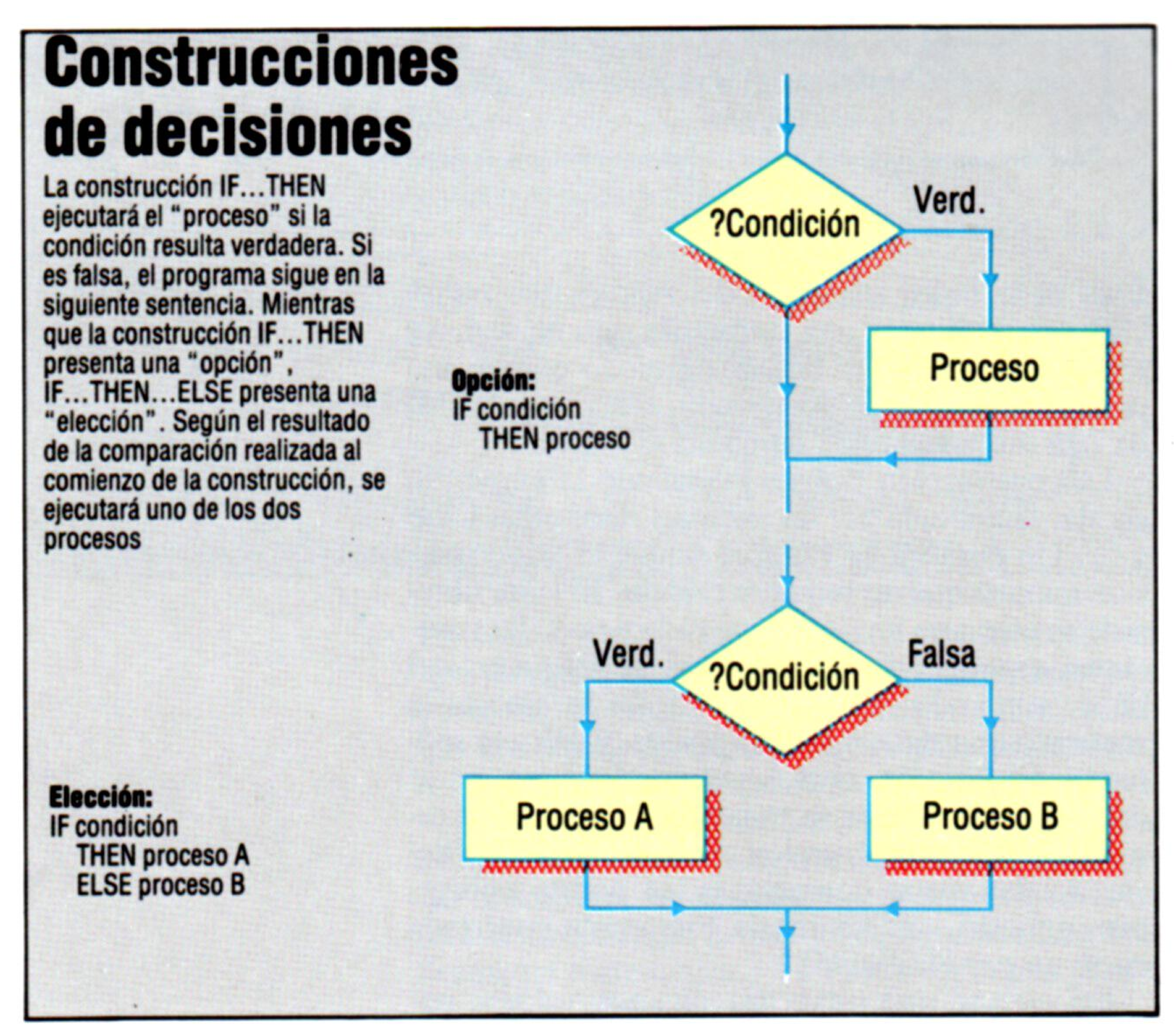
```
CASE N MOD 4 OF
   0 :{no hacer nada};
   1,3: begin
          write (N MOD 4 : 1,'cuarto');
          if N MOD 4>1 then
             write ('s')
        end;
   2 :write ('y medio')
END {CASE}
```

Observe que sigue siendo necesario un punto y coma para separar esta sentencia inexistente de lo que sigue a continuación. La sentencia de la última etiqueta no requiere ninguno porque va seguida de una palabra reservada (END), no por otra etiqueta o sentencia.

El empleo del operador MOD en la expresión asegura que el valor debe estar en la escala entre 0 y 3. MOD da el resto de una división entre enteros, como en el BASIC BBC y otros.

En el próximo capítulo de la serie examinaremos estos y todos los otros operadores del PASCAL, así como las funciones incorporadas. Mientras tanto, he aquí la solución al problema de cómo imprimir las series de caracteres para cada valor de nuestro tipo de color y tinte:

```
CASE color OF
   rojo          :write ('Rojo');
   verde         :write ('Verde');
   amarillo      :write ('Amarillo');
   azul          :write ('Azul');
   magenta       :write ('Magenta');
   cyan          :write ('Cyan');
END {CASE}
```



Construcciones de decisiones

La construcción IF...THEN ejecutará el "proceso" si la condición resulta verdadera. Si es falsa, el programa sigue en la siguiente sentencia. Mientras que la construcción IF...THEN presenta una "opción", IF...THEN...ELSE presenta una "elección". Según el resultado de la comparación realizada al comienzo de la construcción, se ejecutará uno de los dos procesos

Sentencias mensuales

Este programa, que simplemente calcula la cantidad de días entre una fecha entrada por el usuario y el final del mes, ofrece algunos útiles ejemplos de los principios de la programación en PASCAL. La sentencia CASE se emplea para asociar datos de entrada numéricos a los tipos enumerados del PASCAL, utilizándolos para el proceso interno y luego volviéndolos a asociar a datos de salida en serie. Observe el empleo de la variable *char* ficticia (símbolo) para leer los caracteres delimitadores que separan los números entrados

```
PROGRAM              Fecha        (input, output);

CONST
  PuntoFinal='.';

TYPE
    Calendario=(Ene, Feb, Mar, Abr, May,
          Jun, Jul, Ago, Sep, Oct, Nov, Dic);

VAR
    NombreMes          :Calendario;
    dia,
    mes,
    año,
    Faltan             :integer;
    simbolo            :char;
    AñoBisiesto        :boolean;

BEGIN
  WriteLn ('Entre la fecha en la forma  :');
  WriteLn ('DD/MM/AA' : 40 );
  WriteLn;
  write ('Fecha ?');
  read (dia,simbolo,mes,simbolo,año);
  AñoBisiesto:=año MOD 4=0;
                       (ignorando siglos)

  IF (mes>0)AND(mes<=12)
    THEN
      CASE mes OF

        1              :NombreMes :=Ene;
        2              :NombreMes :=Feb;
        3              :NombreMes :=Mar;
        4              :NombreMes :=Abr;
        5              :NombreMes :=May;
        6              :NombreMes :=Jun;
        7              :NombreMes :=Jul;
        8              :NombreMes :=Ago;
        9              :NombreMes :=Sep;
        10             :NombreMes :=Oct;
        11             :NombreMes :=Nov;
        12             :NombreMes :=Dic;

        END            (CASE)
    ELSE
      BEGIN
        WriteLn            ('Eh ?');
        WriteLn            (' - el programa ',
                                'casi se ROMPE !')
      END;                 (NombreMes no está inicializado)

    CASE NombreMes OF

    Ene, Mar,
    May, Jul           : Faltan :=31 - dia;

    Abr, Jun,
    Sep, Nov           : Faltan :=30 - dia;

    Feb                : IF AñoBisiesto
                              THEN
                                  Faltan :=29 - dia
                              ELSE
                                  Faltan :=28 - dia

  END; (CASE)

  WriteLn;
  write ('Faltan ', Faltan : 1,
          'dias ');

  CASE NombreMes OF

            Ene                  :write ('Enero');
            Feb                  :write ('Febrero');
            Mar                  :write ('Marzo');
            Abr                  :write ('Abril');
            May                  :write ('Mayo');
            Jun                  :write ('Junio');
            Jul                  :write ('Julio');
            Ago                  :write ('Agosto');
            Sep                  :write ('Septiembre');
            Oct                  :write ('Octubre');
            Nov                  :write ('Noviembre');
            Dic                  :write ('Diciembre');

  END; (CASE)

  WriteLn (PuntoFinal)

END.
```

Otros juicios

Nuestra comentarista Virginia Rizla hace la valoración crítica de dos programas para el Sinclair Spectrum: uno es de tema deportivo; el otro, de índole "científica"

Daley's decathlon

Ocean

Al haber dejado de lado hace bastante tiempo todo lo que se asemejara siquiera remotamente al ejercicio físico, la perspectiva de tener que jugar al *Daley's decathlon* (El decatlón de Daley) no me causó demasiada alegría, antes bien una acusada sensación de indiferencia.

El objetivo del juego es, naturalmente, competir en las diez pruebas del decathlon (100 m lisos, salto de longitud, lanzamiento de peso, salto de altura, 400 m lisos, 110 m vallas, lanzamiento del disco, salto con pértiga, lanzamiento de jabalina y 1 500 m lisos) y conseguir la máxima cantidad de puntos en cada una. En las pruebas de carrera, el movimiento se controla alternando dos teclas o bien maniobrando la palanca de mando; tras pulsar otra tecla, me encontré saltando enormes distancias y esquivando objetos con notable habilidad y gracia.

Aunque vacilante en la mayor parte del juego, me sentí muy alentada por la clamorosa multitud al conseguir 128 m en el salto de longitud (¡atención a eso, Carl Lewis!). Y no me sentí alienada en lo más mínimo por los convencionales gráficos, por demás realistas, y movimientos.



En resumen, les aconsejo a todos que no boicoteen esta competición.

Pyjarama

Mikro-Gen

Wally tiene una pesadilla y procede a refutar varias de las tesis de Jung sobre el sonambulismo. (Si realmente está inconsciente, parece un poco extraño que el objetivo del juego sea hallar el botón de su reloj despertador.)

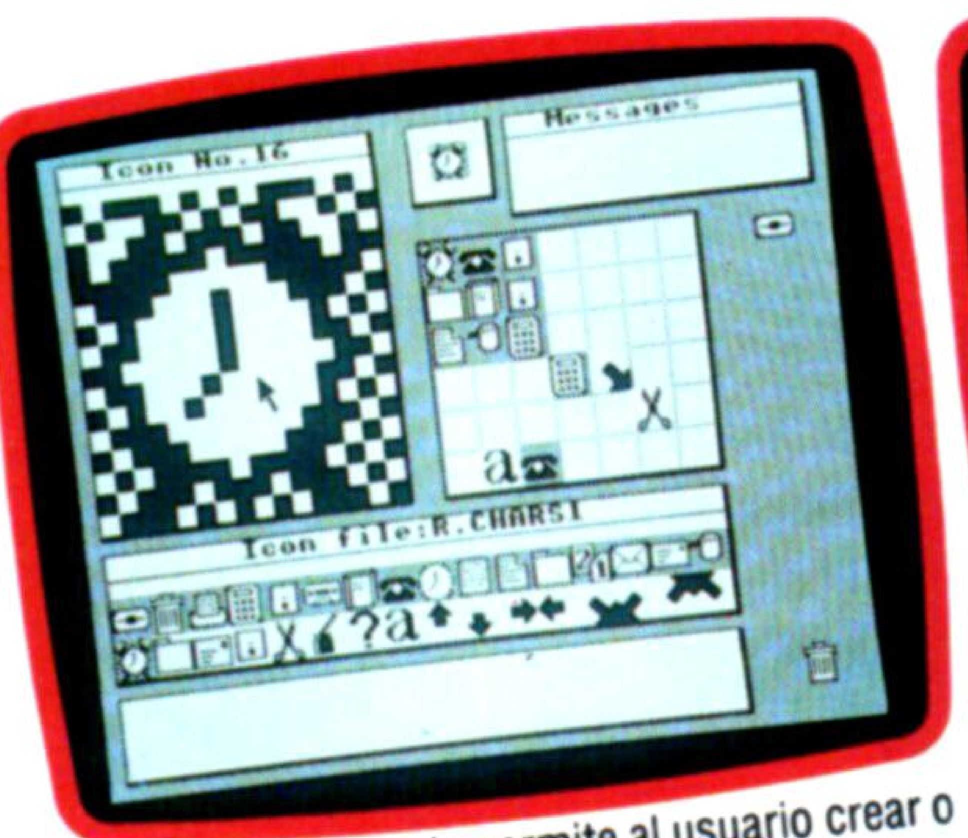
Sin embargo, *Pyjarama* se nos revela como un juego muy entretenido y divertido y no resulta en absoluto irreverente con el psicoanálisis ni las más modernas teorías respecto a la interpretación de los sueños y a las fuerzas que gobiernan nuestro inconsciente.

Varias pantallas representan las habitaciones de la morada de Wally, dibujadas de forma maravillosa y con brillantes colores. Moviéndose a izquierda y derecha y saltando sobre el mobiliario, de dudoso gusto, Wally intenta atravesar las diversas puertas, ya sea saltando para alcanzar el picaporte o gracias a poseer cierto "objeto". Dado que su energía vital es limitada, es preferible evitar los gráficos flotantes y las manos que tienden a aparecer por debajo del suelo.

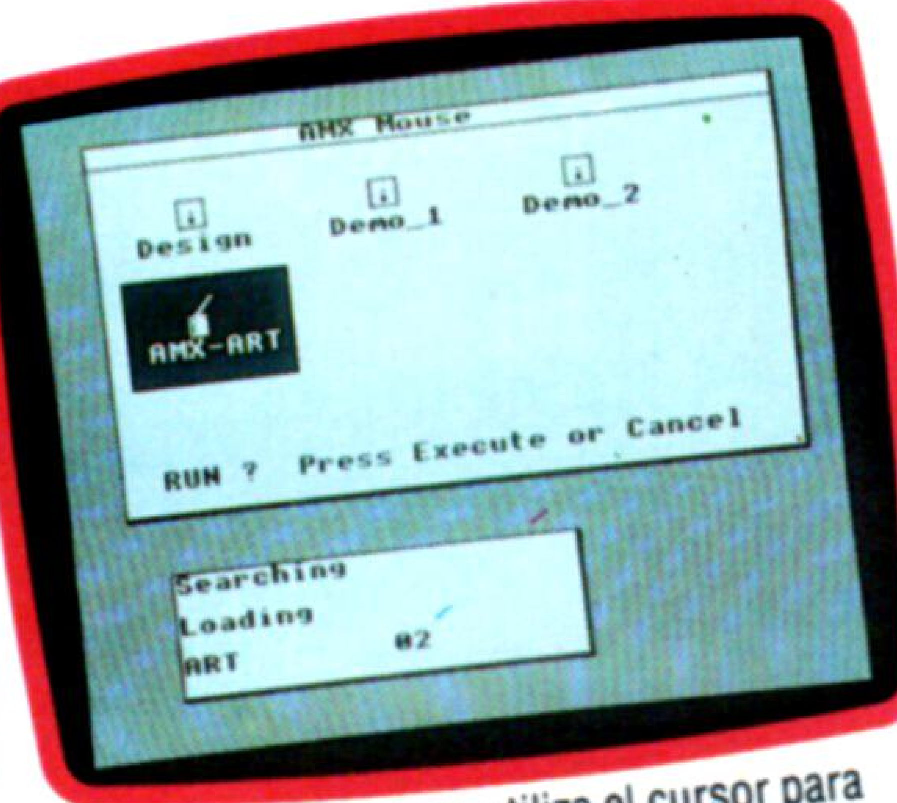


Gran parte del entretenimiento consiste en aprender a jugar al juego, tras lo cual sigue teniendo atractivo visual y, lo que es más importante, interés lúdico.

Arte iconográfico



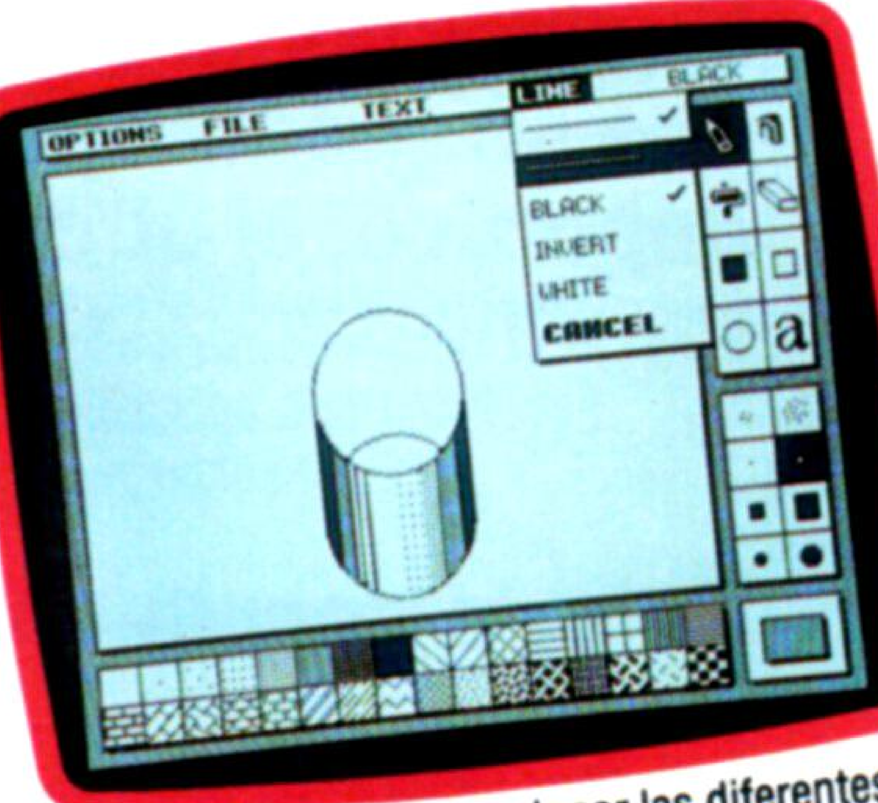
El editor de iconos le permite al usuario crear o adaptar el juego de iconos



El Menú Principal: se utiliza el cursor para seleccionar uno de los cuatro programas



Para crear una imagen: primero se dibuja el contorno



Hay un menú para seleccionar las diferentes líneas disponibles

Analizamos el paquete de ratón AMX, que incluye el programa "AMX Art", creado para el BBC Micro

AMX Mouse
El ratón AMX forma parte de un paquete completo que incluye el ratón propiamente dicho, un chip de ROM que contiene el software activador, software de aplicaciones en disco o cassette, un manual para el usuario y un manual de aplicaciones del *AMX Art*. En cuanto a ejecución, el paquete es muy similar al utilizado en el Apple Macintosh, pero los usuarios del BBC lo tienen a su alcance por una fracción de su precio

Ian McKinnell

El paquete de ratón AMX, producido por Advanced Memory Systems para el BBC Micro, se compone del ratón propiamente dicho, junto con un cable para conectarlo a la puerta para el usuario del micro, un chip de ROM, dos manuales y software en disco y cassette. Si bien se suministran tanto cassette como disco, la propia AMS admite que el AMX está claramente dirigido a quienes poseen unidad de disco.

El AMX, un ratón estándar fabricado en Japón, está moldeado en plástico negro con una bola metálica en su base para la detección del movimiento. El ratón cabe con comodidad en la mano, dejando los tres primeros dedos descansando sobre los tres botones para selección de acción: Move, Execute y Cancel. La decisión de incorporar una bola metálica es poco afortunada, dado que tiende a resbalar muchísimo sobre superficies tales como madera encerada y superficies de escritorio laminadas. Por consiguiente, es una buena idea colocar un trozo de papel sobre cualquier superficie resbaladiza.

En el interior del ratón, la bola está apoyada en dos rodillos; al extremo de cada uno hay un disco con muescas que pasa a través de una célula fotoeléctrica. El disco corta un haz de luz, creando reflejos intermitentes que son captados por la célula, y éstos se transmiten al ordenador como instrucciones de dirección. Por consiguiente, el ordenador conoce con exactitud en todo momento la posición, el movimiento y la velocidad del ratón.

El software residente en la ROM lateral corresponde a la comunicación con el ordenador. Además de proporcionar rutinas "en bruto" para su empleo con software comercial, esta ROM facilita varias rutinas de fácil uso que permiten utilizar el ratón con programas de elaboración casera. Éstas incluyen, entre otras, una rutina para proporcionar una posición actualizada de las coordenadas en pantalla del ratón, otra para leer los botones de selección y una tercera para dar animación a los iconos en la pantalla. Haciendo uso de estas rutinas (todas ellas seleccionadas mediante el empleo de la estructura de instrucción * del sistema operativo del BBC Micro), el usuario puede escribir rápida y fácilmente programas en BASIC que saquen el máximo partido del control del ratón.

También se proporcionan rutinas que permiten el empleo del ratón en paquetes de software comercial ya existentes. El movimiento del ratón se puede configurar de modo que sustituya el uso de las teclas del cursor, y los tres botones de selección para que actúen como tres teclas cualesquiera del teclado, incluyendo las teclas Shift y CTRL. De esta forma es posible, por ejemplo, utilizar el ratón para suprimir, trasladar y copiar secciones de texto en el procesador de textos *View* de Acornsoft.

Sin embargo, la bondad del dispositivo se hace verdaderamente ostensible cuando se utiliza software de aplicaciones escrito especialmente para el



Ian McKinnell



Los contornos se rellenan con varios patrones diferentes...



Y se añaden más texturas



Se agregan palabras y letras pasando a la modalidad de texto y eligiendo el tipo



Mediante la modalidad de mano alzada se dan los toques finales

control por ratón. El paquete AMX incluye dos ejemplos de este tipo de software, que demuestran ampliamente la sencillez y naturalidad del empleo del ratón. El primero es un programa en BASIC para diseñar iconos con el objeto de incorporarlos en los programas creados por el usuario. Los iconos son símbolos gráficos que se pueden visualizar en un programa para indicar las opciones del menú o incluso como sustituto del cursor para indicar la selección efectuada.

El paquete de diseño *Icon* presenta una cuadrícula de 16×16 en la cual usted puede diseñar su icono rellenando los cuadrados deseados: todo ello mediante el ratón. En realidad, con este programa la única ocasión en la que tendrá necesidad de utilizar el teclado será para entrar el nombre de archivo bajo el cual se ha de guardar un grupo de iconos, o el nombre del archivo de un conjunto ya existente que se deba cargar. Debido al enorme control que se ejerce sobre el ratón, éste es sumamente fácil de usar, y diseñar iconos complejos y realistas es una labor rápida y sencilla. No obstante, el paquete *AMX Art*, que también se suministra, ofrece una mejor demostración del potencial del ratón. Éste es un programa para diseño gráfico con características completas y controlado totalmente mediante el ratón. Por su sencillez de uso es comparable al paquete *Macpaint* para el Macintosh y, de hecho, la pantalla del AMX guarda un notable parecido con la creación de Apple.

El programa de arte

El dibujo se efectúa en la zona central de la pantalla; en el programa se utiliza una visualización en Mode 4, de 320 por 256, en dos colores. Sobre el lado derecho de la pantalla hay dos menús de iconos. El menú superior se utilizar para seleccionar diversas acciones de dibujo, incluyendo trazado de líneas, borrado, efectos de pintura con aerosol y rellenado de superficies, que se representan en la pantalla mediante formas fácilmente reconocibles. Una acción se puede seleccionar desplazando el cursor encima del icono en cuestión y pulsando luego uno de los botones de selección, lo cual hace que el cursor cambie de inmediato a la forma del icono seleccionado. Entonces se puede construir la imagen desplazando el cursor a través de la zona de dibujo; mediante el empleo de los botones de selección uno puede llevar a cabo cualquiera de las funciones de dibujo. El excelente manual que se entrega con el *AMX Art* apenas si resulta necesario, debido en parte al uso tan directo del programa y también a que el movimiento del cursor reproduce con gran exactitud el movimiento del ratón.

El menú inferior del lado derecho de la pantalla selecciona la anchura y forma de las líneas de la misma manera. En la parte inferior de la pantalla hay un tercer menú para seleccionar los patrones de sombreado para rellenar las formas, que van desde el negro hasta complejos patrones de enrejado. Puesto que la visualización es en sólo dos colores (por cuestiones de limitación de memoria), se ofrecen estos patrones para diferenciar distintas áreas mediante diversos tipos de sombreado.

Otras opciones, tales como guardar (SAVE) o cargar (LOAD) imágenes, instrucciones del sistema operativo y activación del programa incorporado para vuelcos de pantalla, se pueden seleccionar mediante los menús de la parte superior de la pantalla. Sólo se visualizan los títulos de estos menús, y para revelar el menú completo sobre una sección de su imagen debe posicionarse el cursor sobre uno de los títulos y pulsar uno de los botones de selección. Tras seleccionar sus opciones con el ratón, el menú desaparece dejando la imagen en su forma original.

Desde la primera vez que utilice este programa podrá realizar composiciones complejas, exactas y artísticamente innovadoras con sencillez y rapidez. Tras emplear este paquete durante un breve período, se convierte en un medio tan cómodo como el lápiz y el papel, con la excepción de que el *AMX Art* nunca pierde su filo ni se mancha, y que se puede reproducir tantas veces como se desee.

El programa *Art* ya es de por sí una razón de peso para adquirir el paquete de ratón, pero AMX tiene la intención de ampliar esta primera iniciativa para convertirla en un sistema completo basado en ratón para el BBC Micro. Hay en fase de preparación un gestor de sobremesa de operatoria similar al utilizado en los ordenadores Macintosh y Lisa. También existen planes para una base de datos activada por ratón y más mejoras para el programa *AMX Art*, al cual se le incorporará color y una facilidad de *zoom*.

El ratón AMX y el programa *AMX Art*, así como los otros programas de ratón e iconos que están en preparación, parecen estar llamados a lavarle la cara al BBC Micro, al menos para el usuario que desee ampliar las capacidades de la máquina. Y, por su sencillez de uso, estos programas han establecido un nuevo estándar para los ordenadores personales.

AMX MOUSE

DIMENSIONES
85×65×35 mm

INTERFACES
Un cable a conectar en la puerta para el usuario del BBC Micro

DOCUMENTACION
Dos manuales de fácil comprensión y muy informativos, si bien apenas son necesarios para el paquete *Art*, gracias a la gran amabilidad del paquete hacia el usuario

VENTAJAS
Fácil de operar, manuales excelentes, el paquete *Art* posee muchas capacidades y está bien adaptado al sistema operativo del BBC

DESVENTAJAS
Aparte del paquete *Art*, el software es algo limitado. La bola de metal patina sobre las superficies muy pulidas

¡Todos a bordo!

Iniciamos la creación de un juego basado en una expedición comercial al Nuevo Mundo durante el siglo XVI

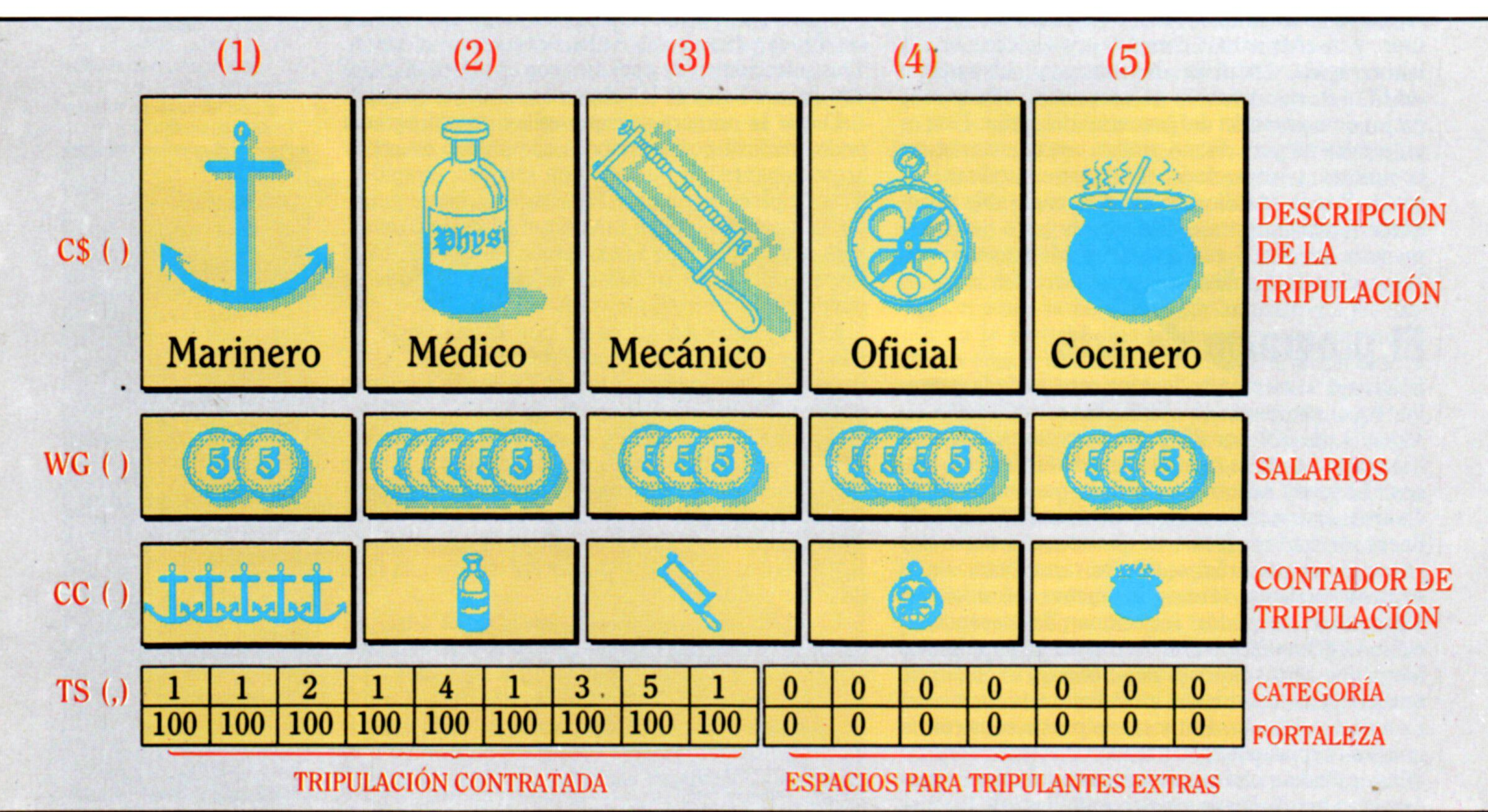
Patrulla de reclutamiento
Se utilizan cuatro matrices para retener los datos relativos a la tripulación contratada para el viaje. C$(), WG() y CC() retienen las características propias de cada categoría de tripulante: descripción, salario y número. TS(,) retiene la composición actual de la tripulación seleccionada para la travesía, hasta un máximo de 16 miembros. Cada una de las categorías de tripulantes puede ser referenciada en función de un número en vez de un nombre, correspondiendo el primero a una dirección de las tres matrices de características de la tripulación

En este juego de simulación usted retrocede en el tiempo y asume el papel de un comerciante del s. XVI que decide organizar una expedición al Nuevo Mundo. Debe planificar el viaje, realizar la travesía y, por último, efectuar transacciones comerciales (de forma rentable, es de esperar) con los nativos.

Al comenzar el juego, el jugador es propietario de un barco mercante y posee 2 000 piezas de oro para gastar. Su primera tarea consiste en contratar una tripulación, ofreciéndole a cada integrante de ella un salario semanal durante la expedición; el monto de los pagos dependerá de la capacidad que posea cada miembro de la tripulación. Ésta debe cubrir una serie de aptitudes, sin que por ello usted se vea obligado a hacer frente a un presupuesto excesivo para la paga de salarios. El navío tiene capacidad para una tripulación compuesta por dieciséis personas; cuanto más numerosa sea, más cara resultará, y consumirá más cantidades de comida y bebida, dejando un menor margen de dinero para las actividades mercantiles. No obstante, si la tripulación fuera demasiado reducida no sería capaz de afrontar algunas eventualidades que pueden producirse durante el viaje, como una epidemia de escorbuto y ataques de piratas, y la travesía no se podría efectuar de forma eficaz, por lo cual la expedición duraría más tiempo y costaría más oro.

El primer módulo trata de la inicialización de variables y matrices que se utilizarán durante el juego y en la contratación de los miembros de la tripulación. La primera sección del módulo DIMensiona las matrices que almacenan importante información acerca de los miembros de la tripulación. C$() se utiliza para retener descripciones de las categorías de tripulantes. Dado que hay cinco categorías de tripulación, la matriz se DIMensiona para tener cinco elementos en la línea 16. Las descripciones se le asignan luego de forma individual a cada elemento de la matriz. Es mucho más conveniente (y ahorra memoria) referenciar las descripciones de la tripulación por su número de elemento en esta matriz. Este número de elemento se conoce algunas veces como la dirección de la matriz, dado que da la posición que ocupa la descripción en la matriz.

El estado de la tripulación varía a lo largo del juego, aumentando o disminuyendo su fortaleza a tenor de las circunstancias. Para llevar un registro de la constitución de la tripulación y de la condición física de cada uno de sus miembros, se utiliza una matriz bidimensional, TS(,). La matriz se DIMensiona en 16 columnas y dos filas. Cada columna representa a un tripulante; la fila superior contiene la categoría que le corresponde y la inferior el coeficiente de fortaleza.



Ian McKinnell



Inicialmente, cada tripulante posee un coeficiente 100 de fortaleza.

El salario de cada categoría de tripulante también se puede representar en una matriz de cinco elementos, WG(). Por consiguiente, si deseamos saber el salario de un médico, categoría de tripulante 2, podemos consultarlo en esta matriz como WG(2). Los marineros reciben 10 piezas de oro por semana, los médicos cobran 25, los oficiales 20 piezas de oro y los cocineros, 15 cada uno. Para llevar el registro de cuánto dinero posee el jugador, en la línea 12 se inicializa una variable, MO, en 2000, que irá decreciendo a medida que se abonen las facturas salariales semanales y se inicien las transacciones comerciales.

Más adelante el programa necesitará saber cuántos tripulantes de cada categoría se han contratado, para poder calcular la eficacia de la tripulación para afrontar las contingencias que puedan producirse. La constitución de la tripulación se podrá determinar, cada vez que se requiera, explorando la fila superior de la matriz TS; pero es más rápido y más fácil preparar otra matriz, CC(), para retener la cantidad de miembros contratados de cada categoría. Nuevamente, esta matriz sólo requiere cinco elementos, y se DIMensiona en la línea 18.

Información preliminar

Tras establecer estas matrices que retienen datos sobre la tripulación, el módulo del programa imprime (PRINT) información relativa a la duración aproximada de la travesía y a la cantidad de dinero que se debe separar para adquirir provisiones. Esta información no se imprime en su totalidad, sino que se emplea una rutina especial para hacer que las letras vayan apareciendo lentamente en la pantalla. Esta subrutina, de la línea 9100, acepta la frase a imprimir en la variable S$ e imprime en la pantalla un carácter por vez, insertando una breve demora para conseguir un efecto de teletipo. Una segunda subrutina, en la línea 9200, inserta pausas más prolongadas, cuando así se requiera, entre la aparición de una línea de texto y la línea siguiente.

La subrutina de la línea 1000 le permite seleccionar su tripulación. Ello se realiza seleccionando una categoría de tripulante entre 1 y 5, o pulsando F para finalizar. Las entradas válidas se almacenan en TS(,) en la línea 1156, llevando CN la cuenta del número de tripulantes contratados hasta el momento. A medida que se va contratando a cada miembro de la tripulación, se visualiza la factura de salarios total por semana, WT, que se va calculando sumándole el elemento correspondiente a la matriz WG() al valor anterior de WT. Esto sucede en la línea 1158. A cada etapa, la sección del programa entre las líneas 1160 y 1190 imprime la composición en curso de la tripulación. La línea 1185 verifica si el contador para una categoría determinada es mayor de uno o cero. De ser así, se añade una S al final de la descripción de las categorías de tripulantes.

En varias etapas del programa se le solicita al jugador que pulse una tecla antes de seguir adelante. La variable K$ se utiliza para retener el mensaje "PULSE CUALQUIER TECLA PARA CONTINUAR" y se transfiere a S$ para su impresión cada vez que se requiera el mensaje. Al retornar de la subrutina de contratación de tripulación, termina el primer módulo.

Módulo Uno

```
   9 K$="  PULSE CUALQUIER TECLA PARA CONTINUAR"
  10 DIM TS(16,2):REM TIPO/FORTALEZA DE LA TRIPULACION
  11 CN=0:REM NUMERO DE TRIPULANTES
  12 M0=2000: REM DINERO INICIAL
  13 DIMWG(5):WG(1)=10:WG(2)=25:WG(3)=15:WG(4)=20:WG(5)=15:REM SALARIOS
  14 WT=0:REM FACTURA SALARIAL SEMANAL
  15 CM=16:REM TRIPULACION MAX
  16 DIMC$(5):C$(1)="MARINERO":C$(2)="MEDICO":C$(3)="MECANICO"
  17 C$(4)="OFICIAL":C$(5)="COCINERO"
  18 DIMCC(5):REM CONTADOR DE CADA TIPO DE TRIPULACION
  80 PRINTCHR$(147):S$="  JUEGO MERCANTIL NUEVO MUNDO*":GOSUB9100:PRINT
  81 GOSUB9200
  82 S$="ERES EL CAPITAN DE UN BARCO*":GOSUB9100:PRINT
  83 S$="QUE SE DIRIGE AL NUEVO MUNDO."SE*":GOSUB9100:PRINT
  84 S$="CALCULA QUE EL VIAJE DURARA OCHO*":GOSUB9100:PRINT
  85 S$="SEMANAS,PERO PODRIA DURAR MAS. DEBES*":GOSUB9100:PRINT
  86 S$="CONTRATAR UNA TRIPULACION, PAGARLES, COMPRAR*";GOSUB9100:PRINT
  87 S$="PROVISIONES, EQUIPO Y MERCANCIAS*":GOSUB9100:PRINT
  88 S$="PARA COMERCIAR. DISPONES DE 2000 PIEZAS*":GOSUB9100:PRINT
  89 S$="DE ORO PARA GASTOS.*":GOSUB9100:PRINT:GOSUB9200
  90 PRINT:S$="          BUENA SUERTE!*":GOSUB9100:GOSUB9200:PRINT
  91 S$=K$:GOSUB9100
  94 GETP$:IFP$=""THEN94
  95 GOSUB9200
 500 GOSUB1000
 999 END
1000 PRINTCHR$(147):PRINT"ETAPA 1 - CONTRATAR LA TRIPULACION"
1010 PRINT"                ----------------------------------
1012 PRINT
1015 GOSUB9200
1020 PRINT:PRINT"CATEGORIAS DE TRIPULANTES DISPONIBLES:"
1025 GOSUB9200
1030 PRINT
1040 PRINT"CATEGORIA  DESCRIPCION       SALARIO SEMANAL"
1050 PRINT"  --------   ----------       --------------"
1060 PRINT"        1    MARINERO        10 PIEZAS DE ORO"
1070 PRINT"        2    MEDICO          25 PIEZAS DE ORO"
1080 PRINT"        3    MECANICO        15 PIEZAS DE ORO"
1090 PRINT"        4    OFICIAL         20 PIEZAS DE ORO"
1100 PRINT"        5    COCINERO        15 PIEZAS DE ORO"
1105 GOSUB9200
1110 PRINT:PRINT:PRINT
1120 S$="ENTRE LA CATEGORIA DE TRIPULANTE REQUERIDA (1-5)*";GOSUB9100
1122 S$="0 'F' PARA FINALIZAR LA CONTRATACION*":GOSUB9100:PRINT:INPUTP$
1125 CT=VAL(P$)
1128 IFLEFT$(P$,1)="F"THENPRINT:PRINT"FIN DE LA CONTRATACION DE
     TRIPULACION.":GOSUB9200:GOTO1310
1130 IFCT>0ANDCT<6THEN1150
1139 PRINT:PRINT
1140 PRINTP$;:S$="  NO ES UNA CATEGORIA DE TRIPULANTE*":GOSUB9100
1142 GOSUB9200
1145 S$="INTENTELO NUEVAMENTE, POR FAVOR"
1146 GOSUB9100
1147 GOTO1300
1150 PRINT:PRINT
1155 CN=CN+1:REM TRIPULACION CONTRATADA HASTA AHORA
1156 TS(CN,1)=CT:REM CATEGORIA DE TRIPULANTE
1157 TS(CN,2)=100:REM FORTALEZA INICIAL
1158 WT=WT+WG(CT):REM SALARIOS TOTALES
1159 CC(CT)=CC(CT)+1:REM CONTADOR CATEGORIAS DE TRIPULACION
1160 S$="TRIPULACION HASTA EL MOMENTO:"
1170 FORT=1TO5
1180 PRINTS$;CC(T);"  ";C$(T);
1185 IFCC(T)>1ORCC(T)=0THENPRINT"S":GOTO1189
1186 PRINT"  "
1189 S$="
1190 NEXT
1195 PRINT:PRINT"FACTURA SALARIAL SEMANAL TOTAL  ";WT
1200 IFCN=CM-1 THENPRINT:S$="SOLO UN TRIPULANTE MAS*":GOSUB9100:GOTO1295
1202 IFCN=CMTHENPRINT:S$="       EL BARCO YA ESTA COMPLETO!!":GOSUB9100;GOTO1310
1295 REM
1300 GOTO1015
1310 PRINT:  S$=K$:GOSUB9100:PRINT:GOSUB9200
1320 GETP$:IFP$=""THEN1320
1999 RETURN
9100 REM IMPRESION LENTA DE S$
9110 FOR S3=1TO32
9115 IFMID$(S$,S3,1)="*"THENS3=32:GOTO9140
9120 PRINTMID$(S$,S3,1);
9130 FORS4=1TO25:NEXT
9140 NEXT:PRINT
9199 RETURN
9200 REM BUCLE DE DEMORA
9210 FORS5=1TO1000:NEXT
9299 RETURN
```

Complementos al BASIC

Spectrum: Introduzca estas modificaciones:

```
16 DIM C$(5,9):C$(1)="MARINERO":C$(2)=
   "MEDICO":C$(3)="MECANICO"
17 C$(4)="OFICIAL":C$(5)="COCINERO"
80 CLS:LET S$="JUEGO MERCANTIL
   NUEVO MUNDO*":GOSUB9100:PRINT
94 LET P$=INKEY$:IF P$="" THEN GO TO 94
1005 CLS: PRINT"ETAPA 1 – CONTRATAR
     TRIPULACION"
1128 IF P$(1 TO 1)="F"THEN PRINT
     "FINALIZADA LA CONTRATACION DE
     TRIPULACION.":GOSUB9200:GOTO1310
9115 IF S$(S3 TO S3)="*" THEN
     S3=32:GOTO9140
9120 PRINT S$(S3 TO S3):
```

BBC Micro: Introduzca estas modificaciones:

```
80 CLS:S$="JUEGO MERCANTIL NUEVO
   MUNDO*":GOSUB9100:PRINT
94 P$=GET$
1005 CLS:PRINT"ETAPA 1 –
     CONTRATACION TRIPULACION"
```

Haciendo los preparativos...
El primer módulo del juego mercantil *Nuevo Mundo* está relacionado con la contratación de la tripulación. Con este fin se establecen varias variables y el jugador puede seleccionar hasta 16 tripulantes

La tortuga va a la escuela

La combinación de BBC Micro y LOGO se está convirtiendo en uno de los sistemas informáticos más utilizados en las escuelas. He aquí dos paquetes escritos para esta máquina

La pantalla de gráficos es de alrededor de 1200 por 800 pasos de tortuga, dando una resolución mejor que la de los paquetes de LOGO para la mayoría de los otros micros. Están disponibles las ocho modalidades de pantalla del BBC Micro, así como las diversas opciones de dibujo (líneas de puntos, relleno triangular, etc.). Si usted utiliza Mode 7, dejará libre muchísima más memoria para aplicaciones que no requieran gráficos. Para sonido, se proporcionan las instrucciones SOUND y ENVELOPE. Éstas funcionan exactamente de la misma forma que lo hacen en el BASIC BBC.

Puede emplear COPY para editar líneas de la forma normal, y puede utilizar los códigos de control del BBC Micro, así como VDU, *FX y otras instrucciones *. Ambas versiones afirman soportar el segundo procesador (6502) y Econet. Ambas funcionaron bien con el Econet, pero no las probamos con el segundo procesador.

Escrita en BCPL y después compilada, la versión de Acornsoft se suministra en dos ROM de 16 K y viene acompañada de tres libros, una cinta y un disco de programas de muestra y ampliaciones del LOGO. Está colmada de características, teniendo más de 200 primitivas, incluyendo las de las ampliaciones en disco. La selección de las primitivas parece haber sido dictada por la motivación de incluir el mayor número posible de las mismas.

Analicemos las características extras más importantes. Un conjunto de primitivas permite la preparación de entornos modificados del LOGO, diseñándolos a la medida para el uso del estudiante. Hay dos características particularmente útiles: las primitivas se pueden redefinir y los procedimientos "enterrar" de modo que parezcan se primitivas; y se pueden preparar archivos que ejecuten un procedimiento de inicialización al cargarse.

"Gran parte de los trabajos iniciales en el campo de la inteligencia artificial utilizaron el LOGO, que guarda una estrecha relación con el LISP, el principal lenguaje para el diseño de sistemas expertos."

Para ayudar en la depuración de los programas, hay una amplia variedad de opciones de seguimiento: se puede rastrear cada línea, cada llamada a un procedimiento o cada primitiva, con la opción de hacer una pausa en cada punto. Esta nueva facilidad es excelente y, asimismo, sirve de ayuda para explicarle al estudiante la lógica de un programa. CATCH y THROW proporcionan una forma de saltar de un procedimiento a otro. Estas primitivas se utilizan fundamentalmente para detectar errores de entrada y, para ayudar en esta tarea, se facilitan otras numerosas primitivas.

"El LOGO posee dos tipos de objetos: 'palabras' y 'listas'. Asimismo, posee operaciones que permiten unir objetos, dividirlos en partes diferenciadas, o examinarlos."

Las ampliaciones que se suministran en disco ofrecen un número de facilidades adicionales. Hay un conjunto de primitivas para tratar con las "propiedades" (el LOGO LSCI Apple las posee), cuya idea deriva del LISP. La creación de múltiples tortugas (hasta 32), capaces cada una de ellas de tener una forma exclusiva y poder direccionarse de modo

"Acornsoft LOGO", de Chris Jobson y John Richards; producido por Acornsoft Ltd, Betjeman House, 104 Hills Road, Cambridge CB2 1LQ, Gran Bretaña; basado en ROM

individual, es factible mediante otro conjunto de primitivas. Hay ampliaciones que permiten impartir instrucciones de tortuga al BBC Buggy, la tortuga Jessop o la tortuga Valiant. Nosotros probamos la ampliación para el Buggy y nos pareció muy fácil de programar.

La versión Logotron se escribió en lenguaje máquina y viene en una única ROM de 16 K, con un manual de formación y de referencias en forma de carpeta de anillas. Fue desarrollada por SOLI (Systèmes d'Ordinateur LOGO International), quien en el pasado trabajó en estrecha relación con LCSI y fue también responsable del LOGO Spectrum. Esta versión es muy similar a cualquier otro LOGO LCSI.

"La característica más importante del LOGO es que puedes hacer que refleje tus intereses, tus necesidades y tu personalidad."

Con alrededor de 120 primitivas, SOLI ha optado de forma deliberada por restringir la cantidad de facilidades disponibles, con el objeto de darles cabida en una sola ROM. Ello supone una economía en el uso de ranuras para ROM y permite, asimismo, una mayor velocidad. La versión Logotron posee numerosas características extras que vale la pena destacar, una de las cuales permite escribir y leer programas en los archivos de texto soportados, al igual que el LOGO Atari. Se ha tenido gran cuidado al diseñar el editor de pantalla. Se han utilizado las teclas de función para controlar el editor y se han incorporado instrucciones FIND y REPLACE.

"Al LOGO se lo considera un 'lenguaje de programación para niños'. Pero también es un 'lenguaje de programación para científicos de ordenadores'."

Se proporcionan dos primitivas nuevas, OPPS y OPNS, que producen listas de nombres de procedimientos y nombres de variables, respectivamente. Éstas se pueden emplear como alternativas para las instrucciones más normales POTS y PONS. Una facilidad muy útil permite pasar variables a las instrucciones del OS BBC. Ésta permite que usted escriba procedimientos de nombre coherente para reemplazar los crípticos mensajes que exige normalmente el OS BBC.

La ausencia de dos primitivas resulta notable: TEXT y DEFINE. En el manual se ofrecen definiciones para ellas, pero dado que operan escribiendo en un archivo y volviéndolo a leer, obviamente son lentas. A los programadores estructurados les agradará saber que, tal como sucede en los LOGO Spectrum y Atari, se ha previsto la falta de espacio mediante la inclusión de una sentencia GO. La instrucción TRACE imprime el nombre de cada procedimiento, así como sus entradas, pero al abandonarlo no se produce ningún mensaje. Hubiera sido muy útil una opción TRACE completa.

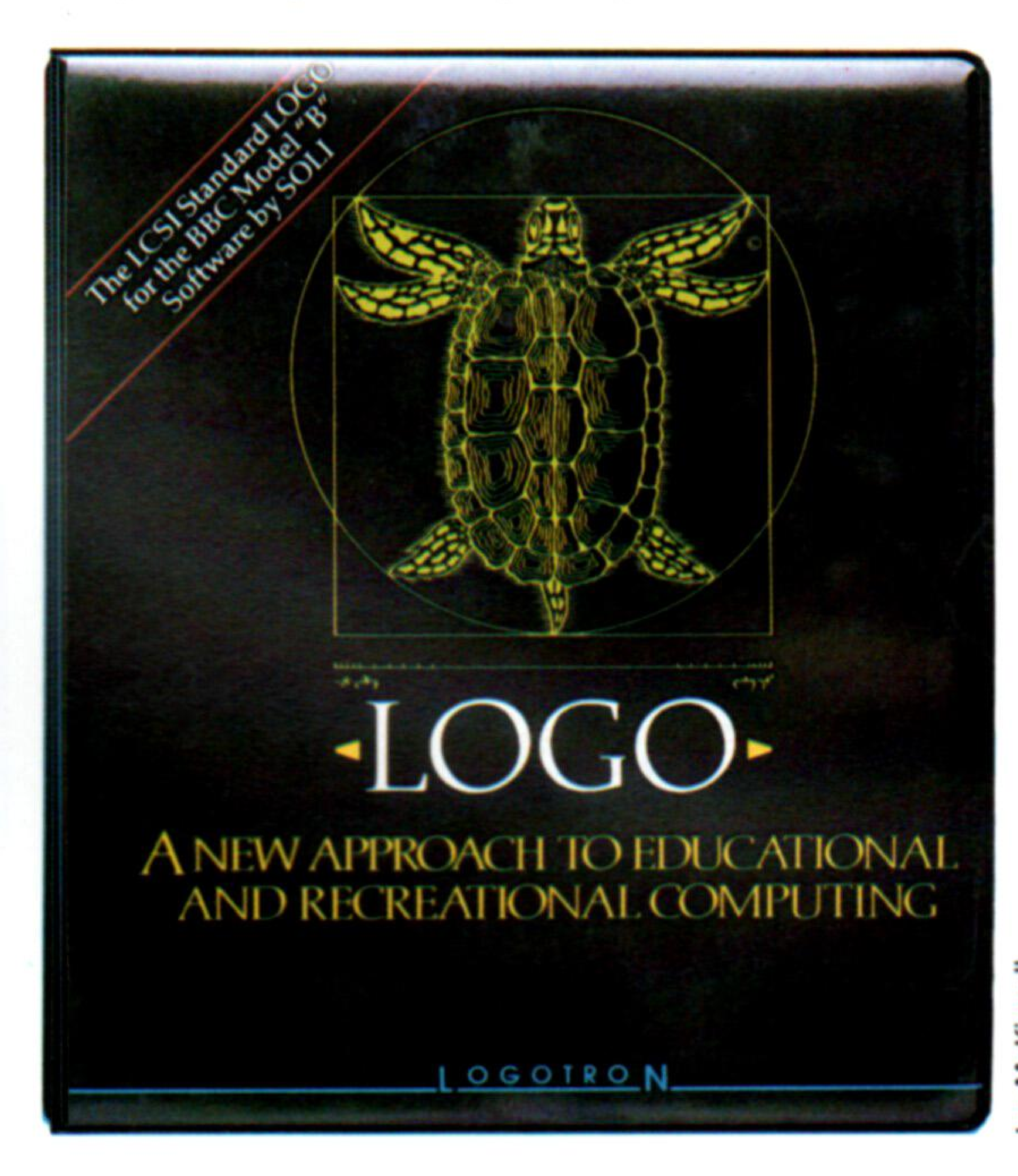
Ejecutamos varios programas de comparación *(benchmark)* con el fin de ofrecer algunas comparaciones entre las dos versiones desde el punto de vista de espacio de trabajo y velocidad. La versión LCSI posee un poco más de espacio de trabajo para definir variables y procedimientos, y alrededor del doble de espacio para la "pila", permitiendo, por consiguiente, más niveles de recursión antes de agotar la memoria. Ambas versiones implementan eficazmente la *recursión sin fin* para instrucciones y operaciones. Esto significa que los procedimientos de recursión sin fin continúan de forma indefinida.

En diversas pruebas de velocidad, la versión de Acornsoft demoró entre un 50 y un 1 000 % más que la versión Logotron, descubriéndose las diferencias más significativas en el proceso de listas y las asignaciones. (En la versión de Acornsoft, el recolector de basuras pareció especialmente lento.) Esta lentitud del proceso de listas de la versión Acornsoft podría resultar muy irritante; sin embargo, un lenguaje lógico similar al PROLOG, escrito en LOGO y que se suministra en el disco, se ejecutó con una velocidad aceptable.

Ambas versiones se entregan con manuales detallados que incluyen secciones de aprendizaje para principiantes y secciones de referencia. Éstas ofrecen explicaciones exhaustivas de geometría de tortuga, aunque en la versión de Acornsoft la distinción entre instrucción y operación no se expone con claridad. Cada uno de los manuales dedica apenas ocho páginas para describir el proceso de listas, pero la versión de Logotron compensa este hecho incluyendo los listados de algunos procedimientos gracias a los cuales a los estudiantes les resultará mucho más fácil aprender el proceso de listas. (Estos procedimientos son el resultado de los trabajos realizados en el Departamento de Inteligencia Artificial de la Universidad de Edimburgo.)

Acornsoft proporciona un buen número de extensos programas de demostración en disco (o cassette), incluyendo un programa tipo PROLOG y un ejemplo de análisis de "lenguaje natural".

Acornsoft ha prometido más ampliaciones para el futuro. Logotron confía en comercializar pronto un "Advanced LOGO" (LOGO avanzado) en disco, que le añadirá al lenguaje muchas primitivas. Más interesante aún es su promesa de una placa de sprites que configurará 30 sprites por hardware.



Ian McKinnell

"Logotron LOGO", de Gerard Dehan, Armen Valian, Marie Paule Deprund y Eric Millet; producido port Logotron Ltd, Ryman House, 59 Markham Street, London SW3, Gran Bretaña; basado en ROM

Intercambios básicos

Veamos ya qué ocurre entre el BASIC y el sistema operativo del BBC Micro

Esta serie de capítulos se abrió con una ideas sobre los sistemas operativos que hablaban del modo en que un OS nos previene contra futuros cambios en el hardware del ordenador. No puede, pues, sorprender que el BASIC utilice tan profusamente las llamadas OS. La instrucción en BASIC SOUND, por ejemplo, genera una llamada a OSWORD con A=7 para producir sonido; igualmente ENVELOPE emplea OSWORD CON A=8. Lo interesante es notar que podemos modificar el modo en que se comporta el BASIC, dentro de unos límites, cambiando el modo en que se comporta el sistema operativo, es decir, alterando los vectores y escribiendo nuestros propios fragmentos de código máquina que reemplacen las rutinas OS.

Asimismo, OSWRCH, OSASCII y OSNEWL se emplean en varias rutinas que acceden a la pantalla. Hay que observar que las rutinas en código máquina que producen visualizaciones de gráficos por medio de rutinas OS no siempre son mucho más rápidas que las instrucciones equivalentes en BASIC. Tenemos aquí una medida de la eficacia del empleo que hace el intérprete del BASIC de las llamadas OS a la hora de interpretar llamadas tales como MOVE, DRAW y PLOT.

Analicemos dos instrucciones BASIC empleadas para llamar desde el BASIC a programas en código máquina, ya sean rutinas del OS, ya sean programas propios. Son las instrucciones USR y CALL.

La llamada USR

El papel que desempeña la instrucción USR es el de ejecutar un fragmento de código máquina y devolver un valor al BASIC que nos dé detalles del estado de los registros A, X e Y después de haber ejecutado el código máquina. También se recaba información sobre el estado del flag de arrastre y del registro indicador de estado del procesador (PSR). Se llama así:

```
result%=USR(direcc%)
```

donde direcc% es la dirección de la rutina que hay que ejecutar. Antes de generar esta llamada se pueden poner valores en las variables A%, X% e Y%, que se pasarán, por medio de la rutina USR, a los registros A,X e Y, respectivamente. El bit menos significativo de la variable C% puede también utilizarse para afectar el valor del flag de arrastre; si C% es impar, éste se pondrá a cero.

El siguiente programa muestra el uso de USR en un breve ejemplo, donde llamamos a una de las llamadas OSBYTE que devuelve un valor en los registros X e Y. Esta rutina da el valor de la abscisa *x* del cursor del texto en el registro X, y el de la ordenada *y* de dicho cursor en el registro Y. El valor contenido en result% representa estos registros en forma codificada, de la que hablaremos luego.

```
10 A%=134
20 X%=0
30 Y%=0
40 LET result%=USR(&FFF4)
50 PRINT result%:REM imprime la representacion
   hexa
```

La impresión en hexadecimal del resultado nos facilita la decodificación de los valores dados por la llamada OSBYTE.

Función USR en acción

La función USR permite al usuario llamar desde el BASIC una sección del código máquina designada para devolver un valor al programa en BASIC. Dado que USR es una función (al contrario de CALL, que es una sentencia) y devuelve un valor, el resultado debe ser asignado dentro del programa en BASIC o a una variable:

```
result%=USR(&FFF4)
```

o como parte de una sentencia PRINT:

```
PRINT USR (&FFF4)
```

El resultado devuelto al BASIC por la función USR toma la forma de un número entero de cuatro bytes que refleja el contenido de los cuatro registros (P,Y,X y A) como sigue:

result%=

Reg. indic. de estado del proces.	Registro Y	Registro X	Registro A

Byte más significativo — Byte menos significativo

Podemos obtener el valor deseado para un registro dado, empleando el operador AND que enmascara ciertas partes del resultado. Por ejemplo:

```
result%AND&000000FF
```

pondrá una máscara a cualquier valor excepto el del registro A

La instrucción CALL

El otro método para llamar programas en código máquina es la instrucción CALL. Ya la hemos empleado en varias ocasiones, en concreto, para llamar a OSWORD y otras rutinas del OS. En esos casos sólo llamábamos a la rutina especificando su dirección. Así,

```
CALL &FFF1
```

El BASIC del BBC permite también una versión ampliada de esta instrucción, que permite el acceso de un programa en código máquina a los valores rete-

nidos en las variables BASIC. Es obvio que el problema principal de tal organización es cómo decir al programa en código máquina dónde puede encontrar las variables en la memoria. Por suerte, el BASIC resuelve este problema gracias al intérprete; si hacemos una CALL con parámetros tal como

```
CALL direcc%,A%,C%
```

todos los detalles sobre los parámetros (en este caso, A% y C%) se almacenan en la página 6 de la memoria. Esta área de la memoria se conoce como *bloque de parámetros*, y el siguiente cuadro ilustra la organización de dicho bloque.

Posición	Descripción del contenido
&0600	Número de parámetros
&0601	Dirección del primer parámetro
&0602	
&0603	Tipo del primer parámetro
&0604	Dirección del segundo parámetro
&0605	
&0606	Tipo del segundo parámetro

Naturalmente, las entradas de la dirección y el tipo se hacen tantas veces como son necesarias. La dirección del parámetro es casi inmediata; se halla donde encontraremos el primer byte del parámetro en la memoria. La entrada del tipo indica qué tipo de variable es el parámetro. Este otro cuadro describe el abanico disponible de tipos de parámetros:

Tipo	Descripción	Ejemplo
0	Byte de ocho bits	CALL direcc,?&70
4	Variable entera	CALL direcc,F%
8	Variable real	CALL direcc,G
128	Serie en una direcc.	CALL direcc,$(&A00)
129	Variable en serie	CALL direcc,F$

La entrada final, tipo de parámetro 129, es ligeramente diferente: no es del todo exacto que la entrada de la dirección en el bloque de parámetros apunta a la posición de la serie en la memoria. De hecho apunta a un área de memoria que nos proporciona más detalles sobre la serie. Esta área de memoria se llama *bloque de información de la serie* y se organiza de esta manera:

Posición	Descripción de contenidos
Dir. contenida en bloque parám.	Dirección de la serie
	Bytes ubicados
	Longitud de la serie

Así, los primeros dos bytes del bloque de información apuntan al primer carácter en la serie. La entrada "bytes ubicados" mostrará la longitud máxima asignada a la serie antes de su longitud efectiva, que se encuentra en el cuarto byte.

El pase de parámetros

La sentencia CALL se emplea para ejecutar una sección de código máquina desde el BASIC, pero, a diferencia de la función USR, no devuelve un resultado. El usuario puede, no obstante, incluir una subrutina que pase los valores del programa en código máquina a las variables BASIC, permitiendo su lectura.

El siguiente programa no sólo da un ejemplo de pase de parámetros por medio de la sentencia CALL, sino que también pone un resultado en la variable entera A%. Además, ilustra el modo en que, al generarse errores en programas en código máquina, éstos puedan ser tratados desde el BASIC. Está escrito para BASIC II.

```
10 DIM C400
20 cero=&70
30 FOR I%=0 TO 2 STEP 2
40 P%=C
50 [OPT I%
60 .code LDA &600
62 CMP #1
64 BNE error
70 LDA &603
80 CMP #4
90 BNE error2
100 LDA &601:STA &70
110 LDA &602:STA &71
120 LDY #0
130 LDA &02:STA(cero),Y:INY
140 LDA &03:STA(cero),Y
160 RTS
170 .error BRK
180 EQUB 254
190 EQUS "Numero incorrecto de
    parametros"
200 EQUB 00
210 .error2 BRK
220 EQUB 253
230 EQUS "Tipos equivocados"
240 EQUB 00
250 ]:NEXT
260 A%=0:CALL code,A%
270 PRINT "Fin de las variables
    BASIC",A%
```

El programa devuelve la dirección del fin de las variables BASIC en una variable entera pasada allí como un parámetro, demostrando así que CALL puede también devolver valores al BASIC. Las líneas 60 a 64 comprueban si sólo hay un parámetro, y si esto no es así se emite un oportuno mensaje de error. Las líneas de la 70 a la 90 comprueban entonces que el parámetro pasado a la rutina es un tipo entero, generándose otro mensaje de error si no es así. De la 100 a la 160 transfieren los datos de las direcciones 2 y 3 a los bytes menos significativos de la variable entera que se pasó como un parámetro. Las líneas de la 170 a la 240 sirven para generar los mensajes de error.

En la línea 260 se hace el CALL; pero antes se pone A% a cero porque el programa en código máquina no altera los bytes más significativos de la variable entera en cuestión.

Después de la llamada, la expresión HIMEM-A% dará el número de bytes que quedan cuando el programa en BASIC y las variables han ocupado su parte de memoria.

Conciencia visual

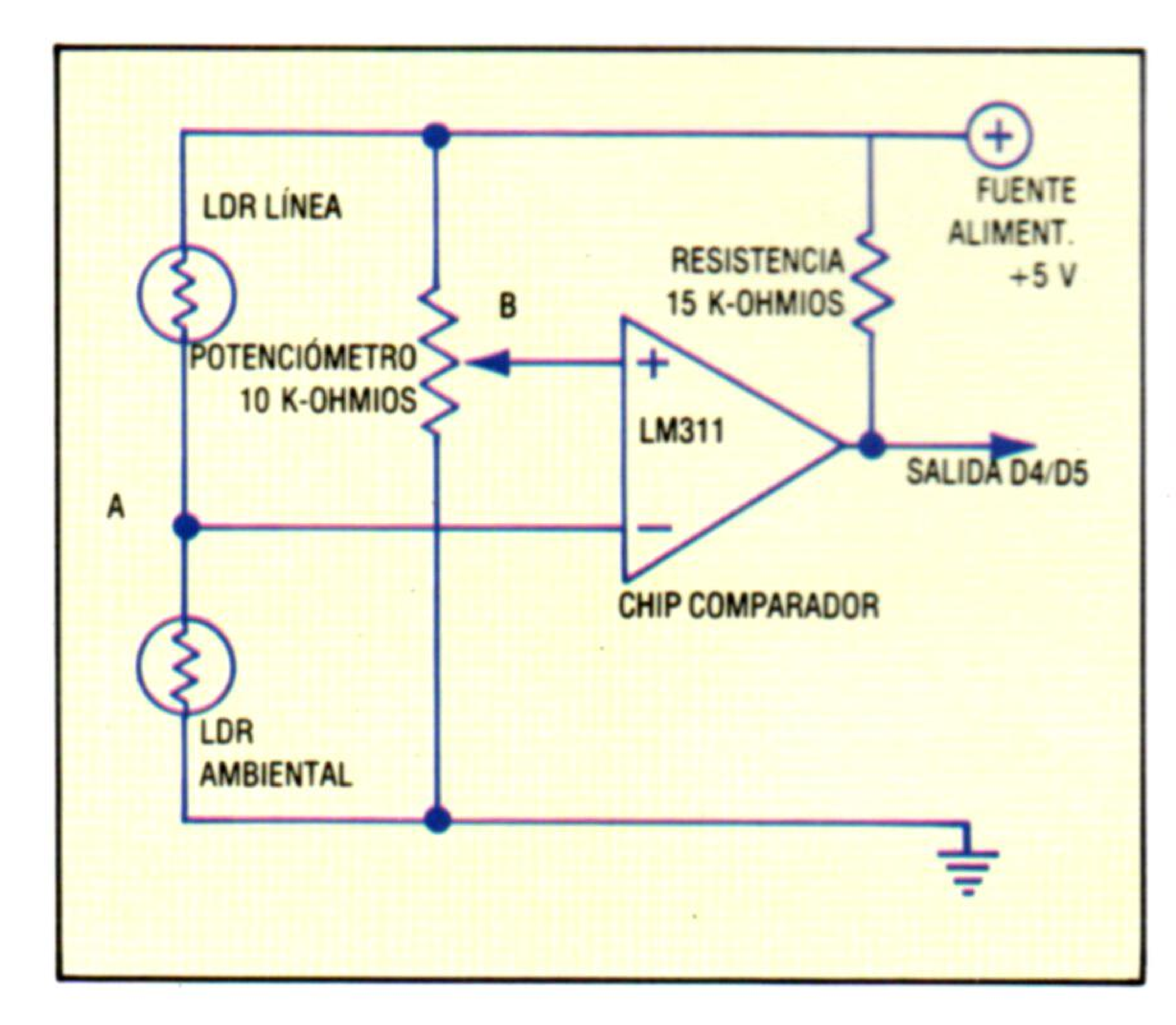
Llegados a este punto, dotaremos al robot con la "vista" necesaria para seguir una línea oscura

La unidad que le permite al robot seguir una línea recta está compuesta por dos circuitos idénticos, alimentando cada uno una línea de entrada diferente de la puerta para el usuario. Cada circuito se basa en un chip comparador LM311, que compara los voltajes de sus dos patillas de entrada. Si el voltaje de la patilla positiva es mayor que el de la entrada negativa, la salida saltará de cero voltios al voltaje de alimentación. En términos lógicos, la salida de este chip será cero si los dos voltajes de entrada son iguales, y uno si el voltaje de la patilla de entrada positiva es mayor que el de la patilla de entrada negativa.

Dos resistencias que dependen de la luz (LDR) en cada circuito forman un divisor de voltaje entre el voltaje de alimentación y tierra. Si la luz que cae sobre los dos LDR es la misma, entonces su resistencia también será igual y el voltaje en el punto A del diagrama de circuito será de unos 2,5 V (la mitad del voltaje de alimentación). El potenciómetro se puede regular para que dé el mismo voltaje en el punto B. Puesto que los dos voltajes de entrada son iguales, la salida del chip comparador será cero.

Si la luz que cae sobre la línea LDR disminuye (p. ej., cuando se halla sobre una línea oscura), su resistencia aumentará, haciendo que el voltaje del punto A sobrepase los 2,5 V. Esto, a su vez, hará que la salida del comparador salte al voltaje de alimentación, dando una salida lógica de uno e indicando que la línea LDR se halla sobre la línea oscura.

La presencia de dos circuitos, compuestos cada uno por un par de LDR y un chip comparador, permite que el ordenador detecte hacia qué lado de la línea se ha inclinado el robot. En cada par de LDR, uno mide la intensidad de la luz ambiental de modo que se pueda comparar con la intensidad de la luz medida por el otro LDR sobre la línea.



Bajando la vista
El circuito seguidor de líneas consta de dos circuitos idénticos, utilizando uno de ellos un par de resistencias que dependen de la luz. Se emplea un chip comparador para producir una señal digital si sobre los LDR están cayendo cantidades diferentes de luz. De esta forma, el robot puede detectar si se halla o no sobre una línea negra

Calibrado

Después de construir e instalar la placa, debemos calibrarla. En condiciones luminosas similares a aquellas en las que usted desea ejecutar el programa seguidor de línea, ejecute este programa de calibrado y regule los dos potenciómetros preestablecidos de la placa recién instalada.

Programas de prueba LDR

```
1000 REM **** PRUEBA LDR BBC ****
1010 RDD=&FE62:REGDAT=&FE60:?RDD=15:REM LINEAS 0-3
     SALIDA
1020 REPEAT
1030 contenido=?REGDAT
1040 IF(contenido AND 16)=0 THEN PRINT TAB(5)"IZQUIERDA";
1050 IF(contenido AND 32)=0 THEN PRINT TAB(15)"DERECHA";
1060 PRINT
1070 UNTIL FALSE
```

```
1000 REM **** PRUEBA LDR CBM ****
1010 RDD=56579:REGDAT=56577:POKE REGDAT,15
1020 CN=PEEK(REGDAT)
1030 IF(CN AND 16)=0 THEN PRINTTAB(5)"IZQUIERDA";
1040 IF(CN AND 32)=0 THEN PRINTTAB(15)"DERECHA";
1050 PRINT
1060 GOTO 1020
```

```
1000 REM **** PRUEBA LDR SPECTRUM ****
1010 CLEAR 32499:LET ST=32500:GO SUB 3000
1020 LET NM=16:GO SUB 2000:IF USR ST=0 THEN PRINT TAB
     5;"IZQUIERDA";
1030 LET NM=32:GO SUB 2000:IF USR ST=0 THEN PRINT TAB
     5;"DERECHA";
1040 PRINT
1050 GO TO 1020
1060 :
2000 REM **** REALIZAR AND ****
2010 POKE ST+1,IN 31
2020 POKE ST+3,NM:RETURN
2030 :
3000 REM **** CARGADOR CODIGO MAQUINA ****
3010 FOR I=ST TO ST+8
3020 READ A:POKE I,A
3030 NEXT I
3040 DATA 62,0,14,0,161,6,0,79,201
3050 RETURN
```

Marque en un trozo de papel blanco una línea negra de 2,5 cm de ancho. Tomando cada potenció-

Lista de componentes

Cantidad	Artículo
2	Chip comparador LM311
2	Conector DIL de 8 patillas
2	Conector 2 mm
2	Resistencia 0,4 W 15 k-ohmios
2	Potenciómetro preestablecido horizontal 10 K
4	LDR ORP12
1	Veroboard 50 agujeros por 24 franjas
1	Rollo parches autoadhesivos
1 m	Cable plano 4 vías
1 m	Alambre estañado pelado 20 swg

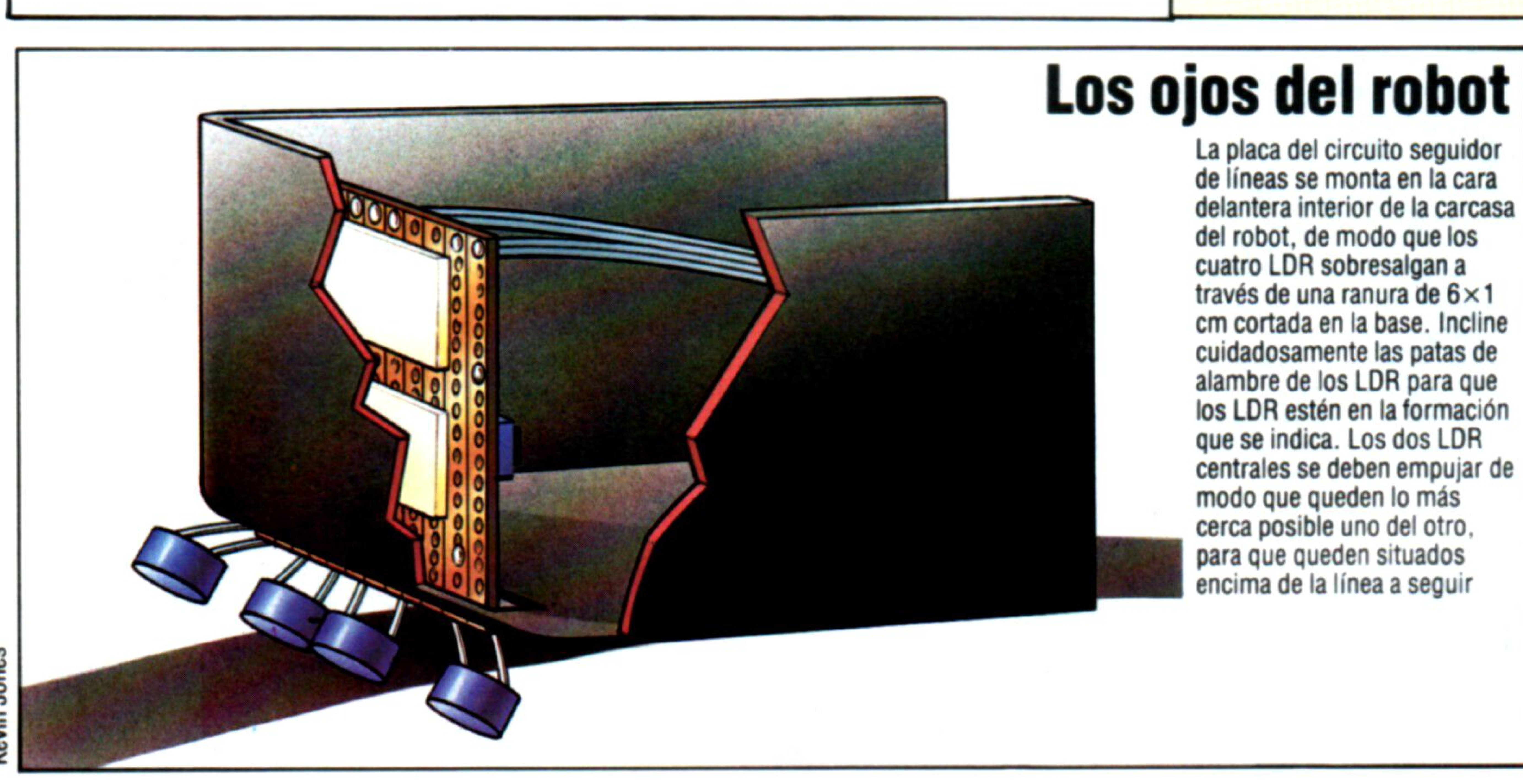


RESISTENCIAS DE 15 K-OHMIOS
LDR
E
SALIDA 1
+5 V
SALIDA 2
+5 V
E
ENCHUFE D DE LA TAPA
SALIDA 1
SALIDA 2
CONECTORES DE PARCHE DE LA TAPA
CABLES DE ENLACE
CHIPS COMPARADORES
POTENCIÓMETROS PREESTABLECIDOS
LDR
SALIDA 2
+5 V
SALIDA 1
E
CORTES DE PISTAS

Trazado de la placa

Corte la veroboard y efectúe los cortes de pistas que se indican. Monte los componentes pasivos, empezando por los cables de enlace y los conectores DIL. Suelde las dos resistencias e inserte los chips comparadores en los conectores DIL, respetando la orientación de las muescas. Suelde los cuatro LDR en el margen de la placa. Deje en los LDR trozos largos de cable para poder situarlos después de que la placa esté colocada. Conecte cuatro cables para enlazar con el sistema de circuitos existente en la tapa del robot. Para incorporar las salidas del chip comparador al sistema de parches, perfore dos nuevos agujeros y monte los conectores detrás del grupo de ocho. Suelde los dos cables de salida en estos conectores. Utilice dos trozos de parche para conectar los conectores de salida de la placa seguidora de líneas a las líneas 4 y 5 del grupo de conectores de parche. Suelde los cables de la alimentación de 5 V y de tierra al enchufe tipo D

Los ojos del robot



La placa del circuito seguidor de líneas se monta en la cara delantera interior de la carcasa del robot, de modo que los cuatro LDR sobresalgan a través de una ranura de 6×1 cm cortada en la base. Incline cuidadosamente las patas de alambre de los LDR para que los LDR estén en la formación que se indica. Los dos LDR centrales se deben empujar de modo que queden lo más cerca posible uno del otro, para que queden situados encima de la línea a seguir

Kevin Jones

metro de uno en uno, ajústelo con un pequeño destornillador de modo que el ordenador no dé ninguna salida cuando los dos LDR centrales se hallen sobre la línea, pero dé la salida correcta cuando el robot se desplace hacia alguno de los lados de la línea. Tras el ajuste compruebe la función desplazando lentamente el robot hacia los lados, a través de la línea, de izquierda a derecha. Cuando los cuatro LDR estén sobre el papel blanco, en la pantalla deberá visualizarse IZQUIERDA DERECHA. Cuando el robot se desplace sobre la línea, el mensaje deberá rezar IZQUIERDA, y cuando no se produzca ningún mensaje el robot estará colocado directamente sobre la línea. Cuando el robot se incline hacia la derecha, la pantalla deberá visualizar DERECHA y, por último, IZQUIERDA DERECHA cuando los cuatro LDR estén otra vez sobre el papel blanco. El programa da por sentado que el LDR de la derecha está parcheado en la línea 4 y el LDR de la izquierda en la línea 5.

Las condiciones de luz operativas deben permanecer constantes durante el calibrado y en las posteriores ocasiones en que se utilice el programa seguidor de línea. Es una buena idea iluminar con intensidad la superficie operativa empleando una luz brillante de flexo, de modo que en futuras ocasiones se puedan reproducir las mismas condiciones luminosas.

Programa seguidor de línea

El programa seguidor de línea utiliza rutinas para mover y hacer girar al robot. El programa da por sentado que éste comienza estando sobre la línea. Emplea un bucle recursivo para desplazar el robot 1 mm hacia adelante y comprobar si se ha apartado de la línea. Esto se detecta cuando el bit 4 o el bit 5 del registro de datos de la puerta para el usuario se pone *low*. Si, supongamos, el bit 4 se pone *low*, ello indica que el robot se ha apartado de la línea hacia la izquierda; el programa, en consecuencia, hace que el robot efectúe un giro de 5° hacia la derecha antes de continuar. Del mismo modo, si el que se pone *low* es el bit 5, es porque el robot se ha desviado hacia la derecha, produciéndose a continuación un giro de 5° hacia la izquierda. El programa termina cuando ambos bits se ponen *low*, como sería el caso cuando el robot llegara al final de la línea.

El ángulo de giro, 5°, se obtuvo a través de la experimentación. Quizá usted deba ajustar la cifra para adecuarla a las líneas que haya dibujado y a sus curvas. Tal vez desee, asimismo, ampliar este programa de modo que el robot encuentre primero la línea y luego la siga. El sistema de parches permite establecer combinaciones de sensores para las cuatro líneas de entrada disponibles en el registro de datos, de modo que usted puede desarrollar sus propias aplicaciones.

Progs. seguidores de línea

BBC Micro

```
10 REM **** SEGUIDOR DE LINEA BBC ****
20 :
30 PROCinicializar
40 PROCseguir_linea
50 END
60 :
70 DEF PROCseguir_linea
80 REPEAT
90 PROCmover(adelante,1)
100 PROCprobar_ldr
110 UNTIL banderafinal=1
120 ENDPROC
130 :
140 DEF PROCprobar_ldr
150 valldr=?REGDAT AND 48
160 IF valldr=32 THEN PROCgirar(derecha,5)
170 IF valldr=16 THEN PROCgirar(izquierda,5)
180 IF valldr=0 THEN banderafinal=1
190 ENDPROC
200 :
210 DEF PROCinicializar
220 RDD=&FE62:REGDAT=&FE60:?RDD=15
230 ?REGDAT=1
240 adelante=4:atras=2:izquierda=6:derecha=0
250 coef_id=3.34446:coef_ia=379/90
260 banderafinal=0
270 ENDPROC
280 :
290 DEF PROCmover(dir,distancia)
300 ?REGDAT=(?REGDAT AND 1)OR dir
310 impulsos=coef_id*ABS(distancia)
320 FOR I=1 TO impulsos:PROCimpulso:NEXT I
330 ENDPROC
340 :
350 DEF PROCgirar(dir,angulo)
360 ?REGDAT=(?REGDAT AND 1)OR dir
370 impulsos=coef_ia*angulo
380 FOR I=1 TO impulsos:PROCimpulso:NEXT I
390 ENDPROC
400 :
410 DEF PROCimpulso
420 ?REGDAT=(?REGDAT OR 8)
430 ?REGDAT=(?REGDAT AND 247)
440 ENDPROC
```

Commodore 64

```
10 REM **** SEGUIDOR DE LINEA CBM ****
20 :
30 GOSUB2000:REM INIC
40 GOSUB1000:REM SEGUIR LINEA
50 END
60 :
1000 REM **** SEGUIR LINEA ****
1010 DR=FW:DS=1:GOSUB2500:REM MOVER
1020 GOSUB1500:REM PROBAR LOS LDR
1030 IF EF<>1 THEN 1000
1040 RETURN
1050 :
1500 REM **** PROBAR LOS LDR ****
1510 LV=PEEK(REGDAT)AND 48
1520 IF LV=32 THEN DR=RT:DS=5:GOSUB3000:REM GIRAR
1530 IF LV=16 THEN DR=LF:DS=5:GOSUB3000:REM GIRAR
1540 IF LV=0 THEN EF=1
1550 RETURN
1560 :
2000 REM **** INIC ****
2010 RDD=56579:REGDAT=56577:POKERDD,15
2020 POKEREGDAT,1
2030 FW=4:BW=2:LF=6:RT=0
2040 PD=3.34446:PA=379/90
2050 EF=0:REM BANDERAFINAL
2060 RETURN
2070 :
2500 REM **** MOVER (DR,DS) ****
2510 POKE REGDAT,(PEEK(REGDAT) AND 1)OR DR
2520 PL=PD*ABS(DS)
2530 FOR I=1 TO PL:GOSUB 3500:NEXT I
2550 RETURN
2560 :
3000 REM **** GIRAR (DR,DS) ****
3010 POKE REGDAT,(PEEK(REGDAT)AND 1)OR DR
3020 PL=PA*ABS(DS)
3030 FOR I=1 TO PL:GOSUB3500:NEXT I
3040 RETURN
3050 :
3500 REM **** IMPULSO ****
3510 POKE REGDAT,PEEK(REGDAT)OR 8
3520 POKE REGDAT,PEEK(REGDAT)AND 247
3530 RETURN
```

Sinclair Spectrum

```
10 REM **** SEGUIDOR DE LINEA SPECTRUM ****
12 REM SUPRIMIR LINEAS 2010,2020,2510,3010 DE LA VERSION CBM
13 REM E INTRODUCIR ESTAS MODIFICACIONES
15 CLEAR 32499:ST=32500:GOSUB4500
1510 LET NM=48:GO SUB 4000: LET LV=USR ST
3510 OUT 31,DR+9
3520 OUT 31,DR+1
4000 REM **** REALIZAR AND ****
4010 POKE ST+1,IN 31
4020 POKE ST+3,NM:RETURN
4500 REM **** CARGADOR DE CODIGO MAQUINA ****
4510 FOR I=ST TO ST+8
4520 READ A:POKE I,A
4530 NEXT I
4540 DATA 62,0,14,0,161,6,0,79,201
4550 RETURN
```

mi computer

Editorial Delta, S.A.



10077
9 788485 822836